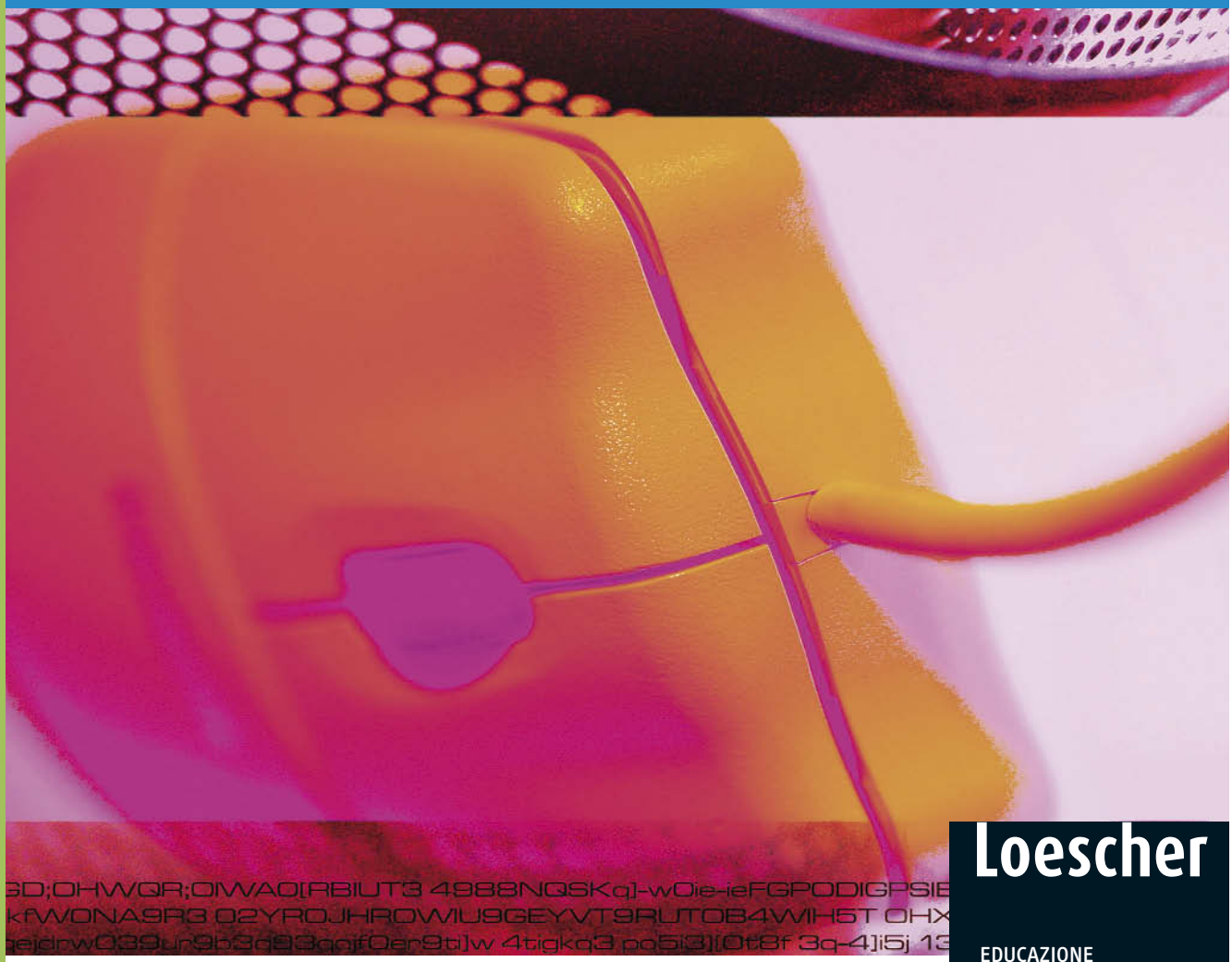
Fabrizia Scorzoni

Informatica

DataBase



Loescher

EDUCAZIONE
TECNICA
SUPERIORE

Fabrizia Scorzoni

Informatica

Database

LOESCHER EDITORE

Loescher Editore - Vietata la vendita e la diffusione

© Loescher Editore - 2009
http://www.loescher.it

I diritti di elaborazione in qualsiasi forma o opera, di memorizzazione anche digitale su supporti di qualsiasi tipo (inclusi magnetici e ottici), di riproduzione e di adattamento totale o parziale con qualsiasi mezzo (compresi i microfilm e le copie fotostatiche), i diritti di noleggio, di prestito e di traduzione sono riservati per tutti i paesi.
L'acquisto della presente copia dell'opera non implica il trasferimento dei suddetti diritti né li esaurisce.

Fotocopie per uso personale (cioè privato e individuale) nei limiti del 15% di ciascun volume possono essere effettuate negli esercizi che aderiscono all'accordo tra SIAE - AIE - SNS e CNA - Confartigianato - CASA - Confcommercio del 18 dicembre 2000, dietro pagamento del compenso previsto in tale accordo; oppure dietro pagamento alla SIAE del compenso previsto dall'art. 68, commi 4 e 5, della legge 22 aprile 1941 n. 633.

Per riproduzioni ad uso non personale l'editore potrà concedere a pagamento l'autorizzazione a riprodurre un numero di pagine non superiore al 15% delle pagine del presente volume. Le richieste per tale tipo di riproduzione vanno inoltrate a:

Associazione Italiana per i Diritti di Riproduzione delle Opere dell'ingegno (AIDRO)
Corso di Porta Romana n. 108, 20122 Milano
e-mail segreteria@aidro.org e sito *web* www.aidro.org

L'editore, per quanto di propria spettanza, considera rare le opere fuori del proprio catalogo editoriale. La riproduzione a mezzo fotocopia degli esemplari di tali opere esistenti nelle biblioteche è consentita, non essendo concorrenziale all'opera.
Non possono considerarsi rare le opere di cui esiste, nel catalogo dell'editore, una successiva edizione, le opere presenti in cataloghi di altri editori o le opere antologiche.

Nel contratto di cessione è esclusa, per biblioteche, istituti di istruzione, musei ed archivi, la facoltà di cui all'art. 71 - per legge diritto d'autore.

Maggiori informazioni sul nostro sito: http://www.loescher.it/fotocopie

Ristampe

7	6	5	4	3	2	1	N
2015	2014	2013	2012	2011	2010	2009	

ISBN 9788884330277

Nonostante la passione e la competenza delle persone coinvolte nella realizzazione di quest'opera, è possibile che in essa siano riscontrabili errori o imprecisioni. Ce ne scusiamo fin d'ora con i lettori e ringraziamo coloro che, contribuendo al miglioramento dell'opera stessa, vorranno segnalarceli al seguente indirizzo:

Loescher Editore s.r.l.
Via Vittorio Amedeo II, 18
10121 Torino
Fax 011 5654200
clienti@loescher.it

Loescher Editore S.r.l. opera con sistema qualità
certificato CERMET n. 1679-A
secondo la norma UNI EN ISO 9001-2000

Coordinamento editoriale: Mauro Gargantini
Revisione: Antonio Giunta - Giuseppe Costa
Collaborazione: Roberto Baldin
Impaginazione: Fabrizia Scorzoni
Disegni: Giorgia Galiazzo

Stampa: Sograte - Città di Castello (PG)

Loescher Editore - Vietata la vendita e la diffusione

INDICE

INTRODUZIONE XII

MODULO 1 - I DATABASE 1

CAPITOLO 1 - INTRODUZIONE AI DATABASE 2

1.1 **Il sistema informativo aziendale e il sistema informatico** 3
1.2 **Database e DBMS** 3
1.3 **Modelli di DBMS** 5
Modello relazionale 6
Modello gerarchico 7
Modello reticolare 8
Database a oggetti 8
1.4 **Progettazione di database** 9
1.5 **Livelli di astrazione** 9
1.6 **Linguaggi per database** 10
1.7 **Modalità di accesso ai database** 11
1.8 **Utenti di database** 12
1.9 **XML** 12

CAPITOLO 2 - DATABASE RELAZIONALI 16

2.1 **Relazioni** 17
Definizione di relazione come concetto matematico 18
Chiave di una relazione 19
Chiavi esterne e legami tra le relazioni 20
Vincoli di integrità 21
2.2 **Algebra relazionale** 22
Unione, intersezione e differenza 23
Selezione 23
Proiezione 24
Join 24
Esempi di espressioni in algebra relazionale 29
2.3 **Viste** 31

CAPITOLO 3 - SQL 36

3.1 **Il linguaggio SQL** 37
Standard relativi a SQL 37
3.2 **Primi elementi** 38

Loescher Editore - Vietata la vendita e la diffusione

Definizione di nuovi tipi di dato ... 39
3.3 Operazioni di definizione dei dati ... 40
Creazione di tabelle ... 40
Vincoli intrarelazionali predefiniti ... 41
Vincolo interrelazionale di integrità referenziale ... 41
***Asserzioni* ... 42**
Modifica di tabelle ... 43
Eliminazione di tabelle ... 43
3.4 Definizione degli indici ... 43
3.5 Manipolazione dei dati ... 44
3.6 Interrogazioni ... 45
Formato base del comando SELECT ... 45
Query parametriche ... 47
La condizione di ricerca ... 47
Ordinamenti ... 49
Funzioni di aggregazione ... 49
Raggruppamenti ... 51
Operazione di join ... 52
Traduzione in SQL di espressioni in algebra relazionale ... 53
Sintassi del join in SQL-2 ... 54
Interrogazioni nidificate o sottoquery ... 56
Unione, intersezione e differenza ... 60
Uso di interrogazioni in comandi di manipolazione ... 62
3.7 Viste ... 62
3.8 Trigger ... 64
3.9 Procedure ... 65
3.10 Controllo dell'accesso ... 65
Ruoli in SQL-3 ... 66
3.11 Le transazioni ... 66
Proprietà ACID delle transazioni ... 67

MODULO 2 - LA PROGETTAZIONE DI DATABASE ... 73

CAPITOLO 4 - IL MODELLO CONCETTUALE ... 74
4.1 Modello ER e UML ... 75
Principali diagrammi UML ... 75
4.2 Entità ... 78
4.3 Attributi ... 79
Dominio degli attributi ... 79
Cardinalità degli attributi ... 80
Attributi derivati ... 80
Identificatori e chiavi ... 81
4.4 Associazioni ... 83
Cardinalità o molteplicità delle associazioni ... 86
Associazioni totali o parziali, univoche o multiple ... 90
Regole di lettura ... 90
Ruoli delle entità nelle associazioni ... 92
Associazioni ricorsive ... 93
Generalizzazioni ... 93
Identificatore esterno ... 95

Loescher Editore - Vietata la vendita e la diffusione

Entità forti e entità deboli ... 96
4.5 Regole aziendali ... 97
Vincoli di integrità ... 97
4.6 Principi di modellazione ... 98
Entità e attributi ... 98
Associazioni con attributi e entità ... 99
Associazioni multiple ... 101
4.7 Esempi svolti ... 102

Capitolo 5 - Il modello logico ... 112
5.1 La progettazione logica ... 113
5.2 Ristrutturazione dello schema concettuale ... 113
Eliminazione delle generalizzazioni ... 113
Partizionamento ed accorpamento di entità ... 115
Ridondanze ... 115
Scelta degli identificatori principali ... 116
5.3 Dallo schema concettuale allo schema relazionale ... 116
Entità e attributi ... 116
Associazioni ... 117
Associazioni ternarie ... 121
5.4 Esempi svolti ... 125
5.5 La normalizzazione ... 127
Anomalie di inserimento, cancellazione e aggiornamento dati ... 127
Dipendenze funzionali ... 128
Prima forma normale ... 129
Seconda forma normale ... 129
Terza forma normale ... 130
Forma normale di Boyce-Codd ... 131
Relazione tra la BCNF e la 3NF ... 135

Capitolo 6 - Il modello fisico ... 143
6.1 Gestione delle operazioni su disco ... 144
Gestione del buffer ... 144
6.2 Strutture primarie e secondarie ... 145
6.3 Organizzazione dei file ... 145
Organizzazione sequenziale ... 145
Accesso calcolato ... 146
Strutture ad albero ... 147
6.4 Progettazione fisica ... 149
6.5 Il gestore delle interrogazioni ... 150

MODULO 3 - DBMS ... 153

Capitolo 7 - MySQL ... 154
7.1 Il DBMS MySQL ... 155
Versioni di MySQL ... 155
Installazione ... 156
Avvio ... 156
Client ... 157

Loescher Editore - Vietata la vendita e la diffusione

7.2 Primi elementi ... 160
Tipi di dato in MySQL ... 160
Operatori e funzioni ... 162
Comandi ... 165
7.3 Definizione dei dati ... 165
SHOW CREATE TABLE ... 165
Motori di memorizzazione ... 165
Campi contatore ... 165
Indici ... 166
Chiavi esterne e integrità referenziale ... 166
Tabelle temporanee ... 167
7.4 Manipolazione dei dati ... 168
Output dei comandi di manipolazione ... 168
Caricamento di dati da un file di testo ... 168
7.5 Interrogazioni ... 168
Output delle query ... 168
Esportazione di dati ... 169
Variabili ... 170
Query parametriche ... 171
7.6 Viste ... 171
***Costrutti per il controllo di flusso* ... 172**
7.7 Trigger ... 172
7.8 Stored routine ... 175
Funzioni ... 175
Procedure ... 176
7.9 Controllo dell'accesso (grant) ... 178
7.10 Transazioni ... 178
***MySQL GUI Tools* ... 179**
7.11 MySQL Query Browser ... 179
La connessione ... 179
L'editor di query ... 180
L'area dei risultati ... 181
La barra di stato ... 181
I box a destra ... 181
Esportazione ... 182
7.12 MySQL Administrator ... 182
Informazioni, avvio e parametri ... 182
Utenti e privilegi ... 183
Funzioni di diagnostica ... 184
Backup e Restore ... 184
Catalogs ... 185
7.13 phpMyAdmin ... 185
Introduzione ... 185
Accesso e pagina principale ... 185
Operazioni sul database ... 186
Operazioni sulle tabelle ... 187
Interrogazioni sul database ... 188

CAPITOLO 8 - ACCESS ... 192
8.1 Il DBMS Access ... 193
Versioni di Access e formato dei file ... 193

Loescher Editore - Vietata la vendita e la diffusione

Interfaccia di Access 2007 ... 193
Apertura o creazione di un database ... 194
Backup di un database ... 195
Esportazione di dati ... 195
Documentazione del database ... 195
Creazione di oggetti ... 195
8.2 Tabelle ... 197
Creazione di tabelle ... 197
Modifica di tabelle ... 197
Proprietà di tabella ... 197
Definizione dei campi ... 198
Definizione della chiave ... 201
Indici e chiavi multicampo ... 202
Il foglio dati ... 203
Elenchi di valori ... 205
Ricerche, ordinamenti e filtri ... 206
Relazioni ... 208
8.3 Query ... 210
Creazione di query ... 210
Creazione guidata query ... 210
Esecuzione della query ... 211
Impostazione di proprietà delle query ... 211
Definizione di query ... 211
SQL ... 214
Query con calcoli e raggruppamenti ... 214
Query a campi incrociati ... 216
Query basate su più tabelle e linee di join ... 217
Query come viste ... 218
Query di comando ... 219
Query di definizione dati ... 220
8.4 Maschere ... 221
Creazione di maschere ... 221
Creazione guidata di una maschera ... 222
Proprietà della maschera ... 222
Stampa di una maschera ... 223
Operazioni sui record tramite le maschere ... 223
I controlli ... 224
Controlli e Proprietà ... 228
Maschere con sottomaschere ... 229
8.5 Report ... 232
Creazione di report ... 232
Creazione guidata di un report ... 233
I controlli ... 233
Le sezioni ... 234
Ordinamento e raggruppamento di dati ... 234
Sottoreport ... 237
8.6 Macro ... 237
Creazione ed esecuzione di una macro ... 237
Azioni non attendibili ... 238
Esecuzione passo a passo ... 239
Gruppi macro ... 239

Loescher Editore - Vietata la vendita e la diffusione

Tasti di scelta rapida ... 240
Gli eventi ... 240
8.7 Codice VBA ... 241
Routine evento ... 242
Confronto tra l'uso di macro e di routine evento ... 242
I moduli ... 243
Gli oggetti di Access ... 244
8.8 Le Applicazioni ... 247
Impostazioni di avvio ... 247
Il pannello di comandi ... 248
Il formato .accde ... 249
Protezione mediante password ... 249
Condivisione di database ... 249

MODULO 4 - ACCESSO A DATABASE ... 255

CAPITOLO 9 - ACCESSO AI DATABASE DALLE APPLICAZIONI ... 256
9.1 Tecnologie di accesso ai database ... 257
Driver per database ODBC (Open DataBase Connectivity) ... 257
Creazione di una origine dati ODBC ... 257
OLE DB ... 258
ADO ... 258
DAO (Data Access Object) ... 259
ADO.NET ... 259
9.2 Accesso a database da Visual Basic con ADO.NET ... 259
Connessione al database e modalità di lavoro ... 259
DbCommand ... 260
DataSet e DataAdapter ... 261
Data binding ... 262
9.3 Accesso a database da Java: driver JDBC ... 266
9.4 Creazione di applicazioni con JDBC ... 267
Caricamento del driver e creazione della connessione ... 267
Operazioni sul database ... 267
Interrogazioni ... 267
Aggiornamenti ... 268
Uso di parametri ... 270
Paginazione e aggiornamento con i metodi di ResultSet ... 271
I metadati ... 272

CAPITOLO 10 - ACCESSO A DATABASE VIA WEB ... 276
10.1 Il protocollo HTTP ... 277
10.2 L'interfaccia CGI ... 277
Le applicazioni server side e l'interfaccia CGI ... 277
Richiesta di un'applicazione server side con un modulo ... 278
Interfaccia CGI per la richiesta ... 278
Richiesta di un'applicazione server side con un collegamento ipertestuale o un indirizzo nella barra indirizzi del browser ... 279
Interfaccia CGI per la risposta ... 279
10.3 Applicazioni server side ... 280

Loescher Editore - Vietata la vendita e la diffusione

Tecnologie server side ... 280
Cookie ... 280
Sessioni ... 281
Architettura 3-tier ... 281
Pubblicazione di un sito ... 282
Server Web ... **282**
DNS ... 283
FTP per la pubblicazione delle pagine ... 283
Server Web accessibile dall'esterno ... 283
Configurazione di Apache ... **283**
Xampp ... 284
10.4 PHP ... **284**
Configurazione ... 284
Primi elementi ... 285
Variabili ... 286
Costanti ... 286
Operatori ... 286
Istruzioni di controllo ... 287
Funzioni ... 288
Array indicizzati numericamente ... 288
Array associativi ... 289
Stringhe ... 289
Gestione dei dati inviati dal browser ... 290
Cookie ... 292
Sessioni ... 292
10.5 Gestione di database in PHP (con MySQL) ... **293**
Collegamento al server ... 293
Visualizzazione di dati ... 294
Inserimento di dati ... 295
Modifica di dati ... 296
Gestione database via ODBC ... 296
Installazione di Tomcat ... **296**
Struttura delle applicazioni Web ... 297
10.6 JSP e servlet ... **297**
Richiesta di pagine JSP ... 298
La prima pagina JSP ... 298
Un esempio di servlet generato dal JSPContainer a partire da una pagina JSP ... 298
Primi elementi ... 300
Oggetti predefiniti ... 301
Request ... 301
Response ... 302
Cookie ... 303
Sessioni ... 304
Gestione degli errori nelle pagine JSP ... 305
10.7 Gestione di database con JDBC ... **306**
Configurazione del server IIS (Internet Information Server) ... **308**
10.8 ASP ... **310**
Oggetti predefiniti ... 310
Response ... 310
Request ... 311

Loescher Editore - Vietata la vendita e la diffusione

Cookie ... 311
Sessioni ... 311
10.9 Gestione di database in ASP con ADO ... 311
Creazione di record ... 313
Modifica di record ... 314
10.10 ASP.NET ... 314
La piattaforma .NET ... 314
Le pagine ASP.NET ... 314
Configurazione ... 315
I primi elementi ... 316
Controlli HTML server ... 316
Form e postback ... 317
Controlli Web server corrispondenti ai controlli HTML ... 318
Posting cross-page ... 319
Controlli di validazione ... 320
Controllo calendar ... 321
10.11 ADO.NET ... 322
DbCommand ... 322
DataSet e DataAdapter ... 323
Data binding ... 324
Controlli data bound ... 325
Controlli data source ... 328
10.12 Sicurezza delle applicazioni server side ... 330
SQL injection ... 330
Mediazione incompleta ed errori tempo di controllo/tempo di utilizzo ... 331

MODULO 5 - XML ... 337

Capitolo 11 - XML e XSL ... 338

11.1 Introduzione ... 339
11.2 Il linguaggio XML ... 339
Applicazioni XML ... 340
11.3 Documenti XML ... 340
Spazi dei nomi ... 343
11.4 Link XML ... 344
XPointer ... 345
11.5 DTD ... 345
Validazione di un documento ... 346
11.6 Fogli stile XSL ... 347
XSL ... 347
Attributi ... 349
XPath ... 351

Capitolo 12 - Elaborazione di documenti XML ... 357

12.1 Java e XML ... 358
API SAX e DOM ... 358
Elaborazione di un documento XML con DOM ... 358
Creazione e modifica dei documenti ... 359
Applicare un foglio stile XSL a un documento XML ... 361
12.2 JSP e XML ... 362

Loescher Editore - Vietata la vendita e la diffusione

Generazione di codice XML ... 362
Elaborazione di documenti XML ... 363
Applicare un foglio stile XSL a un documento XML ... 364
12.3 ASP.NET e XML ... 364
Il controllo Xml ... 364
Lettura e scrittura di documenti XML ... 365
XmlDataSource e TreeView ... 367
DataSet e XML ... 367
DOM XML ... 369

MODULO 6 - PROGETTI ... 375

Capitolo 13 - Temi d'esame ... 376

***Traccia di soluzione* ... 376**
Olimpiadi di informatica ... 377
Trattamento di dati sensibili ... 377
Portale per la vendita di libri ... 378
Community di apprendimento sul Web ... 379
Gestione palestra ... 380
Casa discografica ... 381
Attività scolastiche ed extrascolastiche ... 382
Corsi di inglese online ... 383
Istituto di credito ... 384
Gestione vivaio ... 385
Banca del Tempo ... 386
Band musicale ... 387
Acquisto telematico ... 388
Gestione di un concorso ... 389
Gestione degli ordini ... 389
Servizi turistici "Vacanze" ... 390
Agenzia immobiliare ... 390
Galleria d'arte ... 391
Marketing ... 392
Registro di classe ... 392

Capitolo 14 - Esempi di svolgimento ... 393

14.1 Olimpiadi di informatica ... 393
14.2 Trattamento di dati sensibili ... 406
14.3 Portale per la vendita di libri ... 412
14.4 Community di apprendimento sul Web ... 419

APPENDICE ... 423

Appendice A - Database e diritto ... 424

***Tutela dei dati personali* ... 424**

Indice analitico ... 429

Loescher Editore - Vietata la vendita e la diffusione

INTRODUZIONE

Questo libro si rivolge a tutti i corsi di Informatica in cui si insegnano argomenti relativi ai **database**.

La prima parte (moduli 1 e 2), dopo una breve introduzione generale ai database, tratta gli argomenti teorici relativi ai **database relazionali**, al linguaggio **SQL** e alla **progettazione di database** (concettuale, logica e fisica).

La progettazione concettuale è trattata sia con i diagrammi **ER** che **UML**.

La spiegazione del modello relazionale (capitolo 2) e del linguaggio SQL (capitolo 3) è stata anticipata rispetto agli argomenti di progettazione (capitoli 4, 5 e 6) in modo da consentire in breve tempo di comprendere gli argomenti relativi all'utilizzo di DBMS e alla realizzazione di applicazioni.

Nota

Per chi desidera partire dalla progettazione dei database è sufficiente iniziare dal capitolo 4 (Il modello concettuale) e inserire il capitolo 2 (Database relazionali) prima del capitolo 5 (Il modello logico).

La seconda parte (moduli 3, 4 e 5) è più pratica e tratta:

- l'utilizzo degli ambienti DBMS personali più diffusi: **MySQL** e Microsot **Access**;
- le modalità di **accesso ai database** dalle applicazioni, in particolare utilizzando i linguaggi **Visual Basic** e **Java**;
- le modalità di accesso ai database in rete via Web con le **tecnologie server side** più diffuse: **PHP**, **JSP**, **ASP** e **ASP.NET**;
- il linguaggio **XML** e l'accesso a documenti XML da applicazioni stand alone e server side.

L'ultima parte (modulo 6) presenta i **temi d'esame** proposti negli ultimi anni all'ITIS a indirizzo informatico (Abacus e di ordinamento) e all'ITC a indirizzo programmatori (Mercurio e di ordinamento) con **esempi di svolgimento**.

Per i linguaggi HTML, Visual Basic e Java si rimanda ai volumi:
"**Internet e il WWW**" seconda edizione (codice J020, ISBN 978-88-8433-020-8)
"**Il computer e la programmazione in Visual Basic**" (codice J025, ISBN 978-88-8433-025-3)
"**Programmazione in Java**" (codice J030, ISBN 978-88-8433-030-7)

RINGRAZIAMENTI

Questo libro non sarebbe mai stato finito in tempo senza l'aiuto delle persone che hanno collaborato; un sincero grazie ai colleghi Giuseppe Costa e Antonio Giunta e ai miei giovani collaboratori Roberto Baldin e Giorgia Galiazzo.

L'autrice

Loescher Editore - Vietata la vendita e la diffusione

Modulo 1

I database

1 Introduzione ai database

2 Database relazionali

3 SQL

Loescher Editore - Vietata la vendita e la diffusione

1 INTRODUZIONE AI DATABASE

PREREQUISITI

- Utilizzo di file per la memorizzazione di dati

OBIETTIVI

CONOSCENZE

- Concetti di sistema informativo e sistema informatico, database, DBMS
- Vantaggi dell'approccio database rispetto a quello tradizionale a file
- Concetti di intensione ed estensione di un database
- Modello di DBMS relazionale, gerarchico e reticolare; database a oggetti
- Fasi di progettazione di un database
- Architettura a tre livelli di un database e proprietà di indipendenza logica e fisica dei dati
- Classificazione dei linguaggi per database
- Modalità di accesso a database interattiva, mediante applicazioni e via Web
- Concetto di cursore
- Utenti di un database
- Linguaggio XML

ABILITÀ/CAPACITÀ

- Utilizzare il linguaggio specifico delle basi di dati.

Loescher Editore - Vietata la vendita e la diffusione

1.1 IL SISTEMA INFORMATIVO AZIENDALE E IL SISTEMA INFORMATICO

Ogni azienda tratta molti tipi di **dati**, strutturati come i dati di un dipendente o di un ordine o non strutturati come lettere, articoli, comunicazioni verbali.
I dati sono molto importanti perché da essi si possono ottenere **informazioni**.

Il **sistema informativo** aziendale è l'insieme di tutti i dati utilizzati per lo svolgimento dell'attività dell'azienda e di tutti i procedimenti relativi alla gestione dei dati stessi come la raccolta, l'elaborazione, l'archiviazione e la distribuzione.

La finalità del sistema informativo è la **conoscenza** messa a disposizione delle persone che devono prendere decisioni.

Quando il sistema informativo viene gestito con metodi informatici si parla di **sistema informatico**.

Uno dei principali strumenti del sistema informatico per la gestione dei dati è il **database** o **base di dati**.

1.2 DATABASE E DBMS

Un **database** è l'insieme dei dati utilizzati per rappresentare le informazioni.

Il software di gestione del database è detto **DBMS** (Data Base Management System).

Prima della diffusione dei database i dati erano memorizzati in **file** gestiti da programmi.

L'uso dei database fornisce molti **vantaggi** rispetto ai file.
Un DBMS infatti è in grado di:

- gestire **quantità enormi di dati**, persistenti cioè memorizzati su memoria di massa, garantendo in ogni caso **facilità di accesso**; per la gestione e l'interrogazione dei dati si possono usare linguaggi potenti e specializzati, che possono essere usati direttamente, mediante interfacce grafiche o costruendo applicazioni specifiche;
- garantire l'**indipendenza dei programmi** dal modo in cui i dati sono memorizzati, quindi è possibile modificare le modalità di rappresentazione e registrazione dei dati senza dover modificare anche le applicazioni;
- garantire l'**integrità** cioè il fatto che i dati presenti soddisfino vincoli che garantiscano la significatività dei dati;
- garantire l'**affidabilità** cioè la capacità di conservare o di ricostruire il contenuto anche in presenza di guasti, grazie a meccanismi di salvataggio e ripristino (backup e recovery);
- garantire la **riservatezza** grazie alla gestione di meccanismi che consentono di assegnare autorizzazioni specifiche a ciascun utente;
- permettere la **condivisione dei dati** da parte di più applicazioni e utenti; in questo modo si riduce la **ridondanza** dei dati (cioè le ripetizioni di dati in più punti) e quindi si riduce la possibilità di **inconsistenze** (cioè la possibilità che ci siano copie di dati non uguali).

Per garantire l'accesso condiviso ai dati il DBMS dispone di un meccanismo apposito detto modello di concorrenza.

Nota

Loescher Editore - Vietata la vendita e la diffusione

Tutte le informazioni relative alla descrizione dei dati, ai vincoli da rispettare per garantire l'integrità, alle limitazioni di accesso, e così via, sono memorizzate nel database stesso.
La descrizione dei dati è formata dai **metadati** (dati che descrivono dati) e prende il nome di **dizionario** dei dati o **catalogo** dei dati.

IMPORTANZA DEL CATALOGO

Il fatto che la descrizione dei dati sia conservata nel database rende i programmi indipendenti dai dati e permette al DBMS di effettuare i controlli per l'integrità dei dati e di gestire le autorizzazioni per l'accesso.

L'uso di un DBMS riduce il tempo di **sviluppo delle applicazioni** perché molte funzioni comuni alle applicazioni sono già fornite dal DBMS; inoltre le applicazioni sono più **robuste** perché molti compiti importanti sono gestiti dal DBMS e non devono essere testati in ogni applicazione.

Un DBMS svolge il suo lavoro in modo **efficiente** cioè utilizzando un insieme di risorse (tempo e spazio) ridotto. In realtà ciò dipende dalle tecniche usate per la realizzazione del DBMS e dalla bontà della realizzazione del database da parte dei progettisti.
Il computer su cui il database è installato deve ovviamente essere adeguatamente dimensionato.

Al contrario l'uso dei **file** per la memorizzazione dei dati presenta alcuni **inconvenienti**:

- si può accedere ai file solo mediante i programmi predisposti, cioè solo mediante un numero limitato di operazioni predefinite;
- il tipo di organizzazione del file limita le operazioni che si possono effettuare;
- i programmi dipendono dalla struttura dei file: variare il formato dei record di un file richiede la modifica di tutti i programmi che usano quel file;
- è difficile garantire l'integrità perché le regole che i dati devono rispettare sono esprimibili solo nel codice dei programmi;
- è difficile gestire l'affidabilità, perché si possono usare solo procedure di sistema, come backup periodici;
- è difficile gestire la riservatezza, che viene affidata ai servizi di sicurezza offerti dal file system;
- spesso si creano file diversi per applicazioni diverse con duplicazione dei dati e quindi problemi di inconsistenze.

Nota

Questo problema non deriva dalla diversa natura di file e database ma soprattutto da problemi nella progettazione; errori nella progettazione potrebbero mantenere lo stesso problema anche con i database.

- è difficile gestire la concorrenza nell'accesso ai dati.

L'accesso contemporaneo di più programmi, o di più copie dello stesso programma ad uno stesso file potrebbe causare situazioni di **inconsistenza** dei dati.

Per esempio in una situazione del tipo:
- il programma A legge il record,
- il programma B legge il record,
- il programma A aggiorna il record (per esempio aggiunge 1 a un campo),
- il programma B aggiorna il record (per esempio aggiunge 1 allo stesso campo),

si perde l'aggiornamento del programma A.

I DBMS, per la memorizzazione dei dati, usano file, però con organizzazioni sofisticate.

Loescher Editore - Vietata la vendita e la diffusione

1.3 MODELLI DI DBMS

Nel database sono memorizzati **dati**.
I dati danno **informazioni** solo se si può comprendere il loro **significato** (semantica) o chiave di interpretazione.

Per esempio un numero non dà alcuna informazione se non è specificato se si tratta si un prezzo, un'età, un voto o altro.

Il **significato** dei dati è detto **schema**.

Lo schema deve descrivere l'interpretazione dei dati cioè le classi di dati (**categorie**) definite dalle proprietà che caratterizzano i dati (**attributi**), le relazioni tra le classi (**associazioni**) e i vincoli cui sono soggetti i dati, indipendentemente dai valori che possono assumere i dati stessi.
I **vincoli** possono essere di tipo implicito se sono imposti dalla struttura stessa dei dati, altrimenti sono di tipo esplicito e devono essere opportunamente dichiarati.

Un insieme di dati organizzati secondo lo schema costituisce un **esemplare** (o **istanza**) del database.

L'istanza di un database può cambiare nel tempo; è l'insieme dei dati che interessano in un certo contesto in un certo momento.

Significato intensionale ed estensionale

Lo schema in pratica descrive il significato, l'interpretazione dei dati, detta anche **intensione** (**significato intensionale**); valori specifici dei dati organizzati secondo lo schema costituiscono l'**estensione** (**significato estensionale**).

Il **contenuto informativo** del database deriva sia dal suo significato estensionale che intensionale.

La definizione:

Regioni (Nome, Capoluogo, Popolazione)

costituisce lo schema.

L'insieme dei valori:

Piemonte, Torino, 4287000
Veneto, Venezia, 4512000
Lazio, Roma, 5264000
Sicilia, Palermo, 5087000

costituisce una istanza o estensione.

Su un campo può essere stabilito un vincolo esplicito (per esempio *Popolazione > 0*).

Lo **schema** (categorie, associazioni, ecc.) viene descritto secondo le regole di un **modello** che definisce anche quali sono le operazioni ammesse per accedere ai dati e modificarli.

Un modello è una rappresentazione concettuale che consente di attribuire ai dati una interpretazione del significato e di fornire delle operazioni per la loro gestione.

Un **modello di dati** è costituito da un insieme di regole per la definizione dello schema e da un insieme di operazioni per la gestione della base di dati.

Loescher Editore - Vietata la vendita e la diffusione

Il modello attualmente più usato è quello **relazionale**.
I DBMS relazionali sono anche chiamati **RDBMS** (Relational DBMS).

Modello relazionale

Il **modello relazionale** si basa sul concetto matematico di relazione, nell'ambito della teoria degli insiemi.

Nota: Il modello relazionale è stato definito a livello teorico prima di qualsiasi implementazione sul computer.

Una **relazione** può essere considerata come una tabella di *m* righe e *n* colonne, dove ogni colonna rappresenta un attributo e ogni riga un record.

Un **database relazionale** in pratica è un insieme di tabelle.

Regione

Nome	Capoluogo	Popolazione
Piemonte	Torino	4287000
Veneto	Venezia	4512000
Lazio	Roma	5264000
Sicilia	Palermo	5087000
...	...	...

Città

Nome	Regione
Como	Lombardia
Padova	Veneto
Rovigo	Veneto
Lodi	Lombardia
Messina	Sicilia
...	...

Attraversamento

Fiume	Regione
Po	Piemonte
Po	Lombardia
Po	Veneto
Po	Emilia Romagna
Tevere	Toscana
Tevere	Umbria
Tevere	Lazio
...	...

Fiume

Nome	Lunghezza
Po	652
Adige	410
Tevere	405
Adda	313
...	...

Il modello relazionale è basato sui **valori**: le associazioni tra le entità sono descritte nelle tabelle solo mediante i valori dei campi che compongono le righe delle tabelle.

Nota: Non ci sono archi che collegano le righe delle tabelle.

Le **interrogazioni** possono essere realizzate con un linguaggio formale chiamato **algebra relazionale**, basato su operatori per manipolare le relazioni.
Un linguaggio equivalente è il **calcolo relazionale** basato sulla logica matematica, usata per definire le espressioni che descrivono le proprietà del risultato.

Database relazionale

Un **database relazionale** è un insieme di relazioni (tabelle), variabili nel tempo, le cui istanze sono estensioni di uno schema relazionale.
Uno **schema relazionale** è un insieme di schemi di relazioni.
I legami tra le relazioni si realizzano usando i valori di opportuni campi (campi chiave).

Loescher Editore - Vietata la vendita e la diffusione

Modello gerarchico

Nel **modello gerarchico** lo schema è organizzato in strutture ad albero (alberi di definizione) in cui ogni nodo corrisponde a una entità.
Le associazioni possono essere soltanto gerarchiche: ogni istanza di un nodo è associata ad una e una sola istanza del nodo padre.

Nota

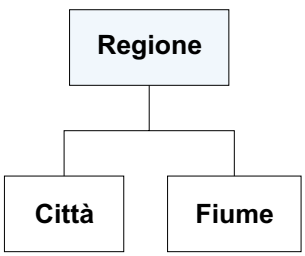
Si parla di associazioni 1:N dal padre ai figli.

Le operazioni di ricerca nell'albero (o navigazione) possono essere compiute solo a partire dalla radice.
Un data base gerarchico risulta efficiente solo se le interrogazioni che vengono poste rispettano la struttura gerarchica.

Domande che non rispettano la struttura possono richiedere la visita di tutto il database.

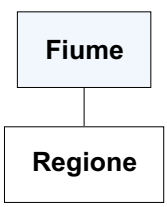
Soluzioni a questo problema richiedono l'introduzione di nuovi alberi, duplicando i dati, o meglio utilizzando figli logici, cioè puntatori ai dati già presenti (che non consentono però un'interpretazione immediata).

Con il database gerarchico costituito da *Regione* come radice e *Città* e *Fiume* come figli è possibile ottenere in modo efficiente risposte a domande del tipo:
Quali sono le città di una data regione?
Quali sono i fiumi di una data regione?



Non è possibile rispondere in modo efficiente a domande del tipo:
Quali sono i le regioni attraversate da un fiume?

Per rispondere a questo tipo di domande può essere necessario un nuovo albero con radice *Fiume* e figlio *Regione*.



In uno schema gerarchico si possono verificare inoltre **anomalie** di inserzione, cancellazione o aggiornamento:

- non si possono inserire informazioni su un nodo figlio se non esiste il padre;
- se si cancella un padre si devono cancellare tutte le informazioni relative ai figli;
- se un dato è duplicato bisogna aggiornarlo in tutti i punti in cui compare.

Nel database dell'esempio precedente si verificano alcune anomalie, per esempio:

- non si possono inserire informazioni su una città se non ci sono informazioni sulla regione a cui la città appartiene;
- se si cancella una regione si devono cancellare anche tutte le informazioni sulle città relative;
- se un fiume attraversa più regioni i dati sul fiume vengono duplicati.

Loescher Editore - Vietata la vendita e la diffusione

Il modello gerarchico è quello che meno realizza l'indipendenza dei programmi dai dati. Per esempio se si inserisce un nuovo livello intermedio nell'albero si devono modificare i programmi.

Nota

Modello reticolare

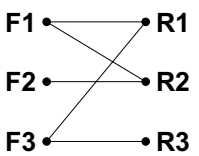
Il **modello reticolare** è una generalizzazione di quello gerarchico. Lo schema è costituito da un grafo orientato (grafo di definizione).

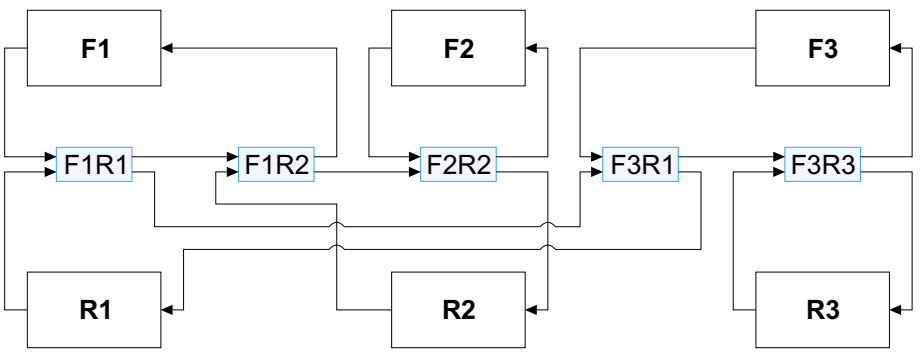
Ogni arco rappresenta una relazione del tipo 1:N.

Nota

Si possono realizzare associazioni tra figli e padre, senza duplicazione di dati, creando i collegamenti tramite categorie fittizie che contengono campi puntatore.

Una associazione tra fiumi e regioni (con il fiume F1 che attraversa le regioni R1 e R2, il fiume F2 che attraversa la regione R2 e il fiume F3 che attraversa le regioni R1 e R3 si può rappresentare nel seguente modo:





Database a oggetti

I **database a oggetti** (**OODB** - Object Oriented DataBase) utilizzano i concetti tipici della programmazione a oggetti: le categorie sono rappresentate da classi ed è possibile sfruttare l'ereditarietà; i dati sono memorizzati come oggetti, quindi possono contenere non solo attributi, ma anche metodi (per esempio metodi che offrono modalità di accesso ai dati).

I database a oggetti sono stati introdotti per l'esigenza di gestire dati multimediali complessi.

I database a oggetti si possono dividere in due tipi:

- gli **OODBMS** (Object Oriented DataBase Management System) estendono i DBMS grazie alle caratteristiche dei linguaggi di programmazione orientati agli oggetti; sono completamente nuovi rispetto ai modelli precedenti;
- gli **ORDBMS** (Object Relational DataBase Management System) integrano il concetto di oggetto all'interno del modello relazionale.

Loescher Editore - Vietata la vendita e la diffusione

1.4 PROGETTAZIONE DI DATABASE

La **progettazione di un database** si divide in progettazione concettuale, progettazione logica e progettazione fisica.

La **progettazione concettuale** definisce una rappresentazione della realtà di interesse, ha come input le specifiche che descrivono il problema e produce come output uno schema concettuale, cioè una rappresentazione astratta, mediante diagrammi.
I diagrammi più utilizzati sono i diagrammi **ER** (Entity Relationship) e quelli **UML** (Unified Modelling Language).

La **progettazione logica** trasforma lo schema concettuale in uno schema logico in base al modello dei dati scelto.
Nel modello relazionale la progettazione logica definisce le relazioni o tabelle.

La **progettazione fisica** definisce tutti gli aspetti che riguardano la rappresentazione dei dati e la loro memorizzazione sul disco.

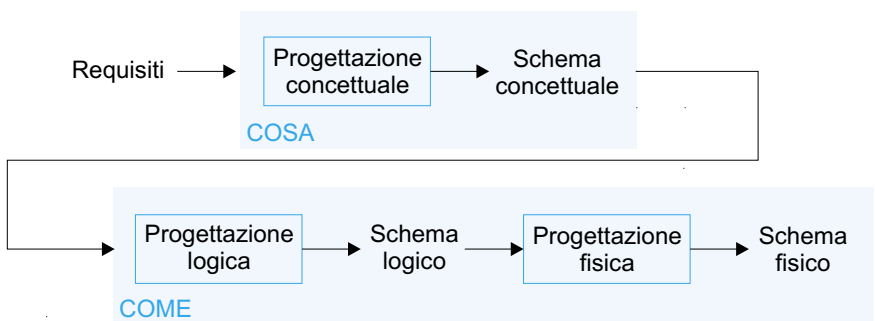


Figura 1.1 *Progettazione di database*

1.5 LIVELLI DI ASTRAZIONE

I dati in un database sono descritti a **tre livelli** di astrazione: **logico** (o concettuale), **fisico** e **esterno**.

La **struttura logica** di un database rappresenta il modo in cui i dati sono organizzati (informazioni e relazioni che intercorrono).
La struttura logica dei dati viene descritta nello **schema logico**, che descrive i dati registrati in termini del modello di dati del DBMS.
In un modello relazionale lo schema logico è costituito dall'insieme delle relazioni o tabelle.
La scelta delle relazioni che compongono lo schema logico avviene durante la fase di progettazione logica.

La **struttura fisica** definisce l'organizzazione dei file a livello fisico sulle memorie di massa.
In questo modo si realizza l'**indipendenza fisica** dei dati: la struttura logica risulta indipendente dal modo in cui è realizzata la struttura fisica ed è possibile modificare lo schema fisico senza dover modificare lo schema logico.

Lo **schema esterno** (chiamato anche **vista** o sottoschema) permette di personalizzare l'accesso ai dati.
Possono essere definite più viste, ciascuna relativa solo ai dati che interessano una particolare applicazione.

Loescher Editore - Vietata la vendita e la diffusione

Le relazioni di una vista possono essere diverse da quelle definite nello schema logico.
Una **vista** è espressa in termini del modello di dati del DBMS; i record della vista non sono memorizzati fisicamente, ma sono ricavati dinamicamente in base alla definizione della vista in termini di relazioni registrate nel DBMS.
È possibile in seguito variare lo schema logico e modificare la definizione della vista in modo che possa essere calcolata come prima, realizzando così l'**indipendenza logica** dei dati.

INDIPENDENZA DEI PROGRAMMI

I programmi sono indipendenti dai dati su entrambi i livelli logico e fisico, nel senso che si possono variare le strutture logiche dei dati senza variare i programmi che li usano e si può variare l'organizzazione a livello fisico senza variare l'organizzazione logica o i programmi.

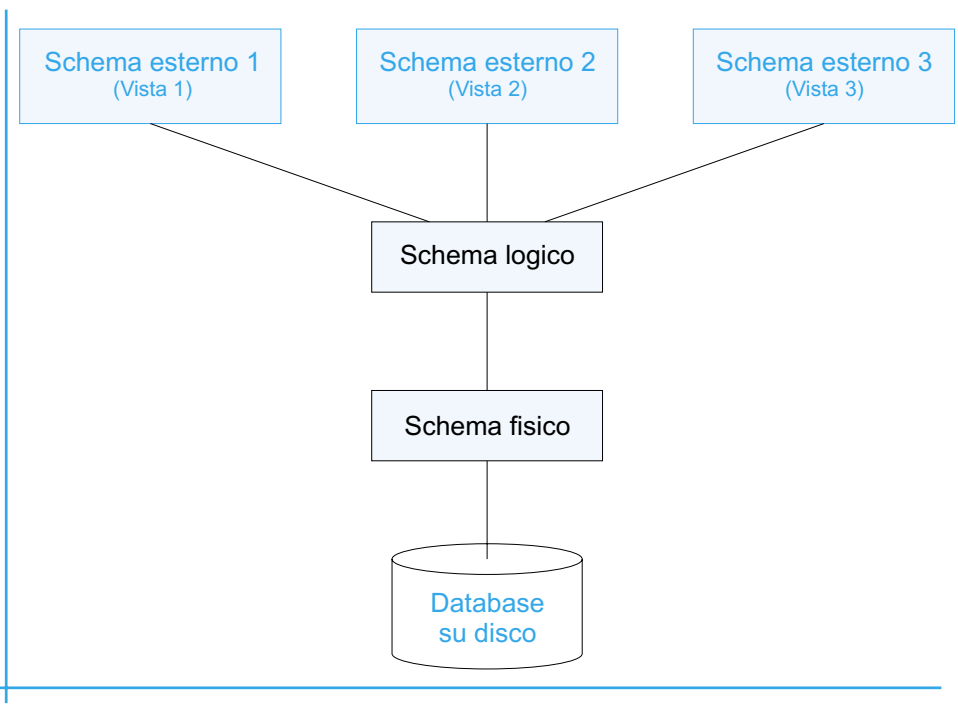


Figura 1.2
Livelli di astrazione dei dati

1.6 LINGUAGGI PER DATABASE

I comandi messi a disposizione dal DBMS per effettuare le diverse operazioni possono essere classificati in diversi tipi di linguaggi in base al loro scopo:

- il **DDL** (Data Definition Language) permette di definire lo schema del database: dati e associazioni con vincoli di integrità e impostazioni di sicurezza e viste;

  I comandi per stabilire i vincoli di integrità e le impostazioni di sicurezza possono costituire un linguaggio a sè chiamato DCL (Data Control Language). Del DDL può far parte anche il DMCL (Device Media Control Language) che permette di definire l'organizzazione dei file sulla memoria di massa (la struttura fisica).

  Nota

- il **DML** (Data Manipulation Language) permette di effettuare operazioni di inserimento, modifica o cancellazione di dati dal database;
- il **QL** (Query Language) è il linguaggio per l'interrogazione del database.

Loescher Editore - Vietata la vendita e la diffusione

Un **linguaggio per database** integra in un unico linguaggio le funzioni di DDL, DML e QL.

I linguaggi per database possono essere di tipo navigazionale o dichiarativo; un linguaggio **navigazionale** è di tipo procedurale e richiede di definire gli algoritmi per l'accesso ai dati; un linguaggio di tipo **dichiarativo** invece permette di accedere ai dati senza conoscere gli algoritmi da utilizzare; per esempio per effettuare una interrogazione basta definire le condizioni che devono soddisfare i dati cercati, senza specificare l'algoritmo di ricerca.

Nota

I linguaggi navigazionali risultano più difficili da usare ma sono più efficienti perché permettono di scegliere i dettagli delle strategie di ricerca.

SQL (Structured Query Language) è un linguaggio per database relazionale di tipo dichiarativo.

QBE

Per effettuare interrogazioni spesso è possibile usare meccanismi di tipo **QBE** (Query By Example - Interrogazione tramite esempio) che richiedono di costruire l'interrogazione riempiendo interattivamente una maschera.

1.7 MODALITÀ DI ACCESSO AI DATABASE

Il linguaggio SQL può essere usato in modo **interattivo** richiamando i comandi di definizione dati, manipolazione, interrogazione e creazione di viste logiche e indici, ma può essere usato anche con modalità diverse.

Per non dover richiamare ogni volta manualmente gli stessi comandi è possibile memorizzare procedure in codice SQL come parte dello schema all'interno del database (**stored procedure**); si ottengono in pratica procedure che possono essere richiamate come i comandi SQL predefiniti.

Per interagire con il database è possibile anche creare **applicazioni** usando linguaggi di programmazione o sfruttando strumenti di sviluppo a interfaccia grafica che permettono di creare applicazioni complete in modo semplice.

I linguaggi di programmazione tradizionali possono essere usati come **linguaggi ospite** dell'SQL, con due modalità:

- **SQL embedded**: nel programma vengono usate delle istruzioni SQL, riconosciute durante la compilazione da un preprocessore che le sostituisce con funzioni di libreria specifiche per ogni sistema,
- **CLI** (Call Level Interface): si utilizzano librerie di funzioni predefinite, specifiche per ciascun linguaggio.

Spesso è possibile utilizzare anche l'**SQL dinamico** che permette di definire le istruzioni SQL in fase di esecuzione, chiedendo all'utente di inserire l'istruzione desiderata.

Nota

Ci devono essere delle modalità per il passaggio di parametri e possono esserci due fasi distinte di preparazione ed esecuzione dell'istruzione.

In generale un'operazione sul database in una applicazione funziona nel modo seguente:

- si crea una connessione con il database;
- si invia un comando SQL;
- si riceve in risposta una tabella di dati su cui è possibile eseguire elaborazioni;
- si chiude la connessione.

Loescher Editore - Vietata la vendita e la diffusione

Un problema per l'integrazione tra SQL e linguaggi di programmazione è il **conflitto di impedenza** (impedence mismatch); SQL è set-oriented, cioè si basa su una visione tabellare dei dati: i comandi operano su una intera tabella e non su una singola riga (per esempio una interrogazione restituisce una tabella); i linguaggi di programmazione invece accedono alle tabelle di dati una riga per volta (sono tuple-oriented).
Per risolvere questo problema vengono utilizzati i **cursori**.

Un **cursore** è uno strumento che indica di volta in volta una sola riga di una tabella; è possibile posizionare il cursore su una riga e usare comandi per spostarsi sulla riga precedente o successiva.

Un **cursore forward only** è un cursore che permette soltanto di avanzare alla riga successiva e che consente solo di leggere le righe ma non di modificarle.

Nota

Nei database orientati agli oggetti il problema del conflitto di impedenza non esiste.

Per consentire un facile accesso ai database sono nati potenti **meccanismi per l'accesso ai dati** che possono essere utilizzati dai linguaggi di programmazione più moderni, per esempio **JDBC** per Java e **ADO** (ora **ADO.NET**) per i linguaggi di programmazione per la piattaforma *Windows*.

Queste tecnologie sono utilizzabili anche nelle **applicazioni server side** e permettono quindi di accedere **via Web**, mediante un browser, a un database che può trovarsi dovunque: su un computer diverso sia da quello del client che da quello su cui si trova l'applicazione server side (**architettura 3-tier**).

Esistono anche **database distribuiti**, cioè composti da varie parti distribuite su vari computer; non è necessario sapere dove si trovano le varie parti per accedere al database.

1.8 UTENTI DI DATABASE

I database vengono creati e utilizzati da molte persone diverse, a vari livelli:

- gli utenti finali o operatori interagiscono con il database mediante programmi applicativi predisposti dai programmatori.
- gli utenti più esperti possono utilizzare comandi SQL o linguaggi QBE;
- i programmatori creano programmi di interfaccia al database usando tutte le tecnologie disponibili;
- l'amministratore del database (**DBA** - DataBase Administrator) si occupa di tutte le problematiche di progettazione, amministrazione e gestione del database (progettazione dello schema e delle viste, progettazione dell'organizzazione fisica, manutenzione e ottimizzazione del database, definizione delle politiche di accesso ai dati da parte di utenti e applicazioni e delle politiche di salvataggio e ripristino e così via).

1.9 XML

In alternativa ai database tradizionali, che richiedono un DBMS proprietario, per la rappresentazione dei dati è possibile usare il linguaggio **XML** (eXtensible Markup Language).

È possibile elaborare file XML con varie modalità ed effettuare trasformazioni da rappresentazioni nel modello relazionale a XML e viceversa.

Loescher Editore - Vietata la vendita e la diffusione

Molti programmi usano XML per memorizzare informazioni di configurazione o come formato standard per lo scambio di dati.

Si può usare un file XML per memorizzare i dati relativi alle regioni:

```
<?xml version="1.0" ?>
<Italia>
    <regione>
        <nome>Piemonte</nome>
        <capoluogo>Torino</capoluogo>
        <popolazione>4287000</popolazione>
    </regione>
    <regione>
        <nome>Veneto</nome>
        <capoluogo>Venezia</capoluogo>
        <popolazione>4512000</popolazione>
    </regione>
</Italia>
```

VERIFICA

QUESTIONARIO

1. Qual è la relazione tra dati e informazioni?
2. Che cos'è il sistema informativo?
3. Che cosa si intende per sistema informatico?
4. Che cos'è un database?
5. Che cos'è un DBMS?
6. Quali sono i vantaggi dell'uso di database rispetto all'uso di file?
7. Che cos'è lo schema di un database?
8. Che cos'è un modello di dati?
9. Su che cosa si basa il modello relazionale?
10. Com'è organizzato lo schema nel modello gerarchico?
11. Com'è organizzato lo schema nel modello reticolare?
12. Che cosa sono i database a oggetti?
13. Quali sono le fasi della progettazione di un database?
14. Quali sono i livelli di astrazione dei dati in un database?
15. Che cosa si intende per indipendenza fisica e indipendenza logica dei dati?
16. Quali tipi di linguaggi possono essere usati per la gestione di un database?
17. Che cos'è SQL?
18. Che cosa si intende per QBE?
19. Quali sono le relazioni tra SQL e linguaggi di programmazione tradizionali?
20. Che cos'è il conflitto di impedenza?
21. Che cosa sono i cursori?
22. Quali meccanismi permettono un facile accesso ai dati di un database?
23. Che cosa sono i database distribuiti?
24. Quali tipi di persone interagiscono con un database?
25. Che cos'è XML?

Loescher Editore - Vietata la vendita e la diffusione

TEST

1 ..... *Indicare se le seguenti affermazioni sono vere o false.*

V	F	
☐	☐	1) I termini dati e informazioni sono sinonimi.
☐	☐	2) Un DBMS è l'insieme dei dati memorizzati nel database.
☐	☐	3) I programmi che usano un database sono indipendenti dal modo in cui i dati sono memorizzati.
☐	☐	4) La proprietà di integrità dei dati di un database garantisce che i dati non possano essere danneggiati da utenti non autorizzati.
☐	☐	5) La proprietà di affidabilità di un database è garantita da meccanismi di backup.
☐	☐	6) Il termine ridondanza indica che gli stessi dati compaiono più volte nel database.
☐	☐	7) Problemi di inconsistenza nel database possono essere dovuti alla ridondanza dei dati.
☐	☐	8) Ridondanze e inconsistenze non sono mai presenti in un database.
☐	☐	9) Un solo utente alla volta può accedere ai dati di un database.
☐	☐	10) Il catalogo dei dati contiene i metadati che descrivono i dati presenti.
☐	☐	11) Lo schema di un database è l'insieme dei dati.
☐	☐	12) Il significato estensionale di un database è l'insieme dei dati.
☐	☐	13) Un modello di dati permette di descrivere lo schema del database.
☐	☐	14) Il modello relazionale è basato su concetti matematici.
☐	☐	15) Un database relazionale fa uso di puntatori per stabilire associazioni tra i dati.
☐	☐	16) Un database gerarchico fa uso di puntatori per stabilire associazioni tra i dati.
☐	☐	17) Un database gerarchico permette di rispondere in modo efficiente a qualsiasi domanda.
☐	☐	18) Una anomalia di inserimento può impedire che alcune informazioni possano essere inserite.
☐	☐	19) Una anomalia di cancellazione può impedire che alcune informazioni possano essere cancellate.
☐	☐	20) Un database reticolare fa uso di puntatori per stabilire associazioni tra i dati.
☐	☐	21) Nei database orientati agli oggetti i dati sono rappresentati da oggetti descritti da classi.
☐	☐	22) Nei database orientati gli oggetti non esiste mai il concetto di relazione del modello relazionale.
☐	☐	23) Lo schema concettuale di un database può essere descritto con un diagramma UML.
☐	☐	24) Lo schema logico di un database può essere descritto con un diagramma UML.
☐	☐	25) Lo schema logico descrive come sono memorizzati i dati sul disco.

Loescher Editore - Vietata la vendita e la diffusione

V	F	
☐	☐	26) Lo schema logico in un database gerarchico è costituito da tabelle.
☐	☐	27) Lo schema logico in un database relazionale è costituito da tabelle.
☐	☐	28) In un database è possibile modificare lo schema fisico senza modificare lo schema logico.
☐	☐	29) Le viste permettono di realizzare l'indipendenza logica dei dati.
☐	☐	30) Si può variare sia lo schema fisico che lo schema logico di un database senza che sia necessario variare i programmi che lo usano.
☐	☐	31) Il DDL è il linguaggio che permette di inserire o modificare i dati nel database.
☐	☐	32) Il QL è il linguaggio di interrogazione del database.
☐	☐	33) Il linguaggio SQL è un linguaggio di tipo dichiarativo.
☐	☐	34) QBE è un linguaggio di tipo procedurale.
☐	☐	35) Il linguaggio SQL può essere usato solo da linea di comando.
☐	☐	36) SQL embedded è un insieme di istruzioni SQL specifiche per un certo linguaggio di programmazione ospite.
☐	☐	37) È possibile accedere a un database via Web.
☐	☐	38) Un database deve sempre trovarsi sullo stesso computer dell'applicazione che lo utilizza.
☐	☐	39) Gli utenti finali di un database di solito usano il linguaggio SQL per interagire col database.
☐	☐	40) I programmatori possono usare il linguaggio SQL per interagire col database.
☐	☐	41) XML è un linguaggio standard per la rappresentazione dei dati.
☐	☐	42) È possibile trasformare in XML la rappresentazione dei dati nel modello relazionale.

2 ..... *Segnare tutte le affermazioni esatte.*

1) Il conflitto di impedenza:

- ☐ è un problema che si può presentare quando si accede a un database relazionale;
- ☐ è un problema che si può presentare quando si accede a un database a oggetti;
- ☐ è dovuto al fatto che i linguaggi di programmazione non possono accedere ai database usando SQL;
- ☐ è dovuto al fatto che SQL vede i dati in modo tabellare;
- ☐ è dovuto al fatto che SQL accede ai dati delle tabelle una riga per volta;
- ☐ può essere risolto usando i cursori.

2) Un cursore:

- ☐ permette di accedere a una riga di una tabella;
- ☐ permette solo di leggere i dati di una tabella, ma non di modificarli;
- ☐ viene utilizzato nei database a oggetti;
- ☐ può essere spostato mediante comandi;
- ☐ può essere di tipo forward only.

Loescher Editore - Vietata la vendita e la diffusione

2 DATABASE RELAZIONALI

PREREQUISITI

- Terminologia generale relativa ai database

OBIETTIVI

CONOSCENZE

- Relazioni, schema relazionale di un database e istanze
- Concetti di chiave, chiave candidata, superchiave, chiave primaria e chiave esterna di una relazione
- Vincoli di integrità intrarelazionali (di chiave, di dominio e su valori) e interrelazionali (integrità referenziale)
- Operatori dell'algebra relazionale: operatori insiemistici (unione, intersezione, differenza, prodotto cartesiano), operatori di selezione e proiezione, operatori di join (join naturale, equi-join, join esterno, self-join e theta-join)
- Concetto di vista

ABILITÀ/CAPACITÀ

- Individuare le chiavi candidate sullo schema di una relazione e fissare la chiave primaria
- Individuare i vincoli di integrità di dominio e su valori su uno schema di relazione
- Individuare i vincoli di integrità referenziale su uno schema relazionale di database e le operazioni che possono causare la violazione dei vincoli di integrità
- Effettuare interrogazioni su una o più relazioni tramite l'algebra relazionale

Loescher Editore - Vietata la vendita e la diffusione

2.1 RELAZIONI

Il **modello relazionale**, il più usato attualmente per i database, è basato sul concetto di relazione.

Una **relazione** può essere considerata come una tabella di *m* righe e *n* colonne.

Nazioni

NomeN	Sigla	Continente	Capitale	Superficie	Popolazione
Italia	I	Europa	Roma	301.333	57.884.000
Stati Uniti	USA	Nord America	Washington	9.629.091	281.422.000
Germania	D	Europa	Berlino	357.030	82.504.000
Francia	F	Europa	Parigi	543.965	60.561.000
Egitto	ET	Africa	Il Cairo	998.002	69.537.000
Canada	CDN	Nord America	Ottawa	9.984.670	32.378.000
Brasile	BR	Sud America	Brasilia	8.514.215	183.552.000
Austria	A	Europa	Vienna	83.871	8.118.000
...	...	...	...	...	...

Ogni colonna della tabella rappresenta un **attributo** di un certo tipo (o campo della relazione); il significato della colonna è spiegato dall'intestazione della colonna stessa. Se è specificata l'intestazione, l'ordine delle colonne non è importante.

Ogni riga della tabella rappresenta un record ed è chiamata **tupla**; le tuple devono essere tutte diverse tra loro e non ha importanza il loro ordine.
Per ogni record è definito un **valore** per ogni attributo. Alcuni attributi sono opzionali e possono permettere che il valore non sia presente (valore ***null***).

Nota

È possibile specificare su quali attributi sono ammessi i valori *null*.

Una relazione è formata da uno **schema relazionale** che descrive le intestazioni delle colonne della tabella e da una **istanza** (o occorrenza) della relazione che è l'insieme dei valori contenuti nella tabella.

Nota

La parola relazione viene usata a volte per intendere lo schema, altre per intendere una istanza.

Lo **schema** di una relazione (**significato intensionale** o intensione della relazione) comprende il nome della relazione e l'elenco degli attributi con il loro tipo.

Lo schema si può rappresentare con la notazione:

NomeRelazione (Attributo1: tipo1, ..., AttributoN: tipoN)

Spesso si evidenzia solo il nome della relazione e degli attributi, omettendo il tipo, descritto a parte.

Nazioni (NomeN, Sigla, Continente, Capitale, Superficie, Popolazione, TipoGoverno, Bandiera, Inno)

Loescher Editore - Vietata la vendita e la diffusione

Una descrizione più chiara di una relazione si ottiene mediante una tabella che elenca nome e tipo dei campi.

Nazioni

Attributo	Formato
NomeN	stringa
Sigla	stringa
Continente	stringa
Capitale	stringa
Superficie	numerico
Popolazione	numerico
TipoGoverno	stringa
Bandiera	immagine
Inno	audio

Identificare gli attributi di una relazione

Si possono identificare gli attributi di una relazione con la **notazione puntata**:

NomeRelazione.NomeCampo

Nazioni (Nome, Sigla, Continente, Capitale, Superficie, Popolazione)
Città (Nome, Nazione, Popolazione)
si possono distinguere gli attributi con lo stesso nome premettendo il nome della relazione: *Nazioni.Nome* e *Città.Nome*, *Nazione.Popolazione* e *Città.Popolazione*.

Si dice **istanza di una relazione** l'insieme delle sue tuple in un determinato momento (**significato estensionale** o estensione della relazione).
L'istanza di una relazione è variabile nel tempo: le tuple possono essere inserite, cancellate o modificate.

Definizione di relazione come concetto matematico

Una relazione è un **concetto matematico**. Il modello relazionale è basato su concetti matematici che permettono di eliminare qualsiasi problema di ambiguità.

Dati n insiemi $D_1, D_2 \ldots D_n$ (con $n \geq 1$) una relazione R è un sottoinsieme del prodotto cartesiano $D_1 \times D_2 \times \ldots \times D_n$.

Ogni tupla della relazione è una ennupla $(d_1\ d_2 \ldots d_n)$ con $d_1 \in D_1, d_2 \in D_2 \ldots d_n \in D_n$.

$$R \subseteq D_1 \times D_2 \times \ldots \times D_n = \{(d_1\ d_2 \ldots d_n) \mid d_1 \in D_1, d_2 \in D_2 \ldots d_n \in D_n\}$$

n si dice **grado** della relazione, gli insiemi D_i sono detti domini (possono esserci più domini uguali); ogni dominio rappresenta un attributo.

Dato che R è un insieme di tuple, in base alla definizione di insieme, non è importante l'ordine delle tuple e non ci possono essere due tuple uguali.

Il numero m di tuple presenti in un certo istante è detto **cardinalità** della relazione.

Loescher Editore - Vietata la vendita e la diffusione

Chiave di una relazione

Si dice **chiave candidata** di una relazione un insieme minimale di attributi che permette di distinguere tra loro le tuple.

Un insieme di attributi è una chiave candidata se gli attributi sono necessari e sufficienti per individuare una tupla, cioè se soddisfa le proprietà:

- **univocità**: i valori dell'insieme di attributi sono sempre diversi al variare delle tuple e quindi permettono di identificare ciascuna tupla;
- **non ridondanza**: l'insieme è il più piccolo possibile (insieme minimale), cioè togliendo uno qualsiasi degli attributi i valori non sono più sempre diversi tra loro.

Per ogni relazione esiste sempre una chiave candidata, al limite costituita da tutti gli attributi della relazione, poiché le tuple sono tutte diverse tra loro.

Nazioni (NomeN, Sigla, Continente, Capitale, Superficie, Popolazione)
Una nazione può essere identificata dal nome (attributo *NomeN*).

Nota: Si suppone che gli stati degli Stati Uniti singolarmente non siano considerati nazioni altrimenti il nome Georgia non identificherebbe univocamente una nazione.

Città (NomeC, Nazione, Popolazione)
Il nome da solo non costituisce una chiave perché possono esserci città con lo stesso nome (per esempio Alessandria in Italia e in Egitto).
Una chiave candidata può essere costituita dagli attributi *NomeC* e *Nazione*.

Un insieme di attributi che soddisfa la proprietà di univocità si dice **superchiave**.

Nota: Una superchiave può non soddisfare la proprietà di non ridondanza: non è un insieme minimale e può contenere attributi non necessari.

In una relazione possono esistere più chiavi candidate; fra queste ne viene scelta una (di solito quella con meno attributi) che è la **chiave primaria**.

Nota: Quando si parla di chiave senza specificare nulla, di solito si intende la chiave primaria.

CodiciPostali (NomeCittà, Codice) (per i capoluoghi di provincia)
Ciascuno degli attributi *NomeCittà* e *Codice* può costituire la chiave; la scelta avviene in base alle interrogazioni più frequenti: se viene cercato più frequentemente il codice dato il nome della città, come chiave viene scelto il nome.

Spesso per motivi di efficienza si fa in modo che la chiave sia un unico attributo di dimensione ridotta, a volte introdotto appositamente (come un codice, o un numero progressivo generato automaticamente).

Città (IdCittà, NomeC, Nazione, Popolazione)
con *IdCittà* numero progressivo che costituisce la chiave.

La chiave primaria non può contenere valori *null*; se è composta da più attributi nessuno degli attributi può contenere valori *null*.

Loescher Editore - Vietata la vendita e la diffusione

Nota

Dopo aver scelto una chiave primaria gli attributi delle altre chiavi candidate possono contenere valori *null*.

Città (IdCittà, NomeC, Nazione, Popolazione)
Scegliendo come chiave *IdCittà* è possibile inserire i dati di una città anche se al momento non si sa a quale nazione appartiene, perché è possibile lasciare *Nazione* a *null*.
Se la chiave primaria è {*NomeC*, *Nazione*} invece non è possibile.

Nello schema di una relazione si devono **evidenziare** gli attributi che formano la **chiave primaria** mediante opportune convenzioni.

Nella notazione

NomeRelazione (Attributo1, Attributo2, ... AttributoN)

si usa la convenzione di sottolineare gli attributi che costituiscono la chiave primaria.

Nazioni (NomeN, Sigla, Continente, Capitale, Superficie, Popolazione)

Nella tabella di descrizione dello schema si può inserire una colonna col valore **PK** (Primary Key).

Nazioni

Attributo	Chiave	Formato
NomeN	**PK**	stringa
Sigla		stringa
Continente		stringa
Capitale		stringa
Superficie		numerico
Popolazione		numerico

Chiavi esterne e legami tra le relazioni

Il legame tra due o più relazioni viene stabilito mediante chiavi esterne.

Un attributo di una relazione è detto **chiave esterna** (foreign key) se può assumere soltanto valori del dominio della chiave primaria di un'altra relazione; se la chiave primaria è composta da più attributi, la chiave esterna è composta dallo stesso numero e tipo di attributi.

Nazioni (NomeN, Continente, Capitale, Superficie, Popolazione)
Continenti (Sigla, NomeCon)
Città (NomeC, Nazione, Popolazione)

Il campo *Nazioni.Continente* deve assumere un valore di *Continenti.Sigla*;
Continente è una chiave esterna per *Nazioni*.

{*Nazioni.Capitale*, *Nazioni.NomeN*} deve assumere un valore di {*Città.NomeC*, *Città.Nazione*};
{*Nazioni.Capitale*, *Nazioni.NomeN*} è una chiave esterna per *Nazioni*.

Poiché la chiave esterna è usata per stabilire un **legame tra le relazioni**, per rendere valide le associazioni il valore della chiave esterna deve essere controllato durante le operazioni di inserimento, cancellazione e modifica delle tuple.

Loescher Editore - Vietata la vendita e la diffusione

Se l'attributo chiave esterna ha un valore che non compare nella chiave dell'altra relazione si possono verificare **anomalie di inserimento**, **cancellazione** o **modifica** di dati collegati tra loro.

> Se è definita la nazione Cile con *Continente* 'SA', ma 'SA' non compare nella relazione *Continenti*, non è possibile sapere qual è il continente corrispondente e se il valore è esatto o errato; inoltre potrebbe essere inserito in modi diversi in altre tuple che dovrebbero avere lo stesso valore (per esempio in Perù potrebbe essere inserito 'AS' o 'AM').

Per garantire l'integrità dei dati, sulle chiavi esterne sono stabiliti **vincoli di integrità referenziale**.

> Se sono stabiliti vincoli di integrità referenziale il controllo di validità della chiave esterna viene fatto in modo automatico.
>
> **Nota**

Vincoli di integrità

Per essere sicuri che i dati presenti nelle relazioni rappresentino informazioni corrette possono essere definiti dei vincoli di integrità.

> I **vincoli di integrità** sono proprietà che devono essere soddisfatte dalle tuple: sono predicati che devono assumere il valore vero perché l'istanza soddisfi il vincolo.

I vincoli di integrità si suddividono in vincoli intrarelazionali e vincoli interrelazionali.

Vincoli intrarelazionali

I **vincoli intrarelazionali** sono definiti ciascuno su una singola relazione:

- **vincoli di chiave** o **integrità di chiave**: sulle chiavi primarie sono vietati valori duplicati e valori *null*;
- **vincoli di tupla**: sono definiti su ciascuna tupla indipendentemente dalle altre; in particolare:
  - un **vincolo di dominio** si riferisce a un solo attributo, imponendo una restrizione sul suo dominio; anche l'obbligatorietà dell'attributo (valore diverso da *null*) è un vincolo di dominio;
  - un **vincolo su valori** coinvolge due o più valori.

> Superficie > 0
> è un vincolo di dominio.
>
> Per la descrizione di una attività:
> DataInizio < DataFine
> è un vincolo su valori.

Vincoli interrelazionali

I **vincoli interrelazionali** coinvolgono più relazioni.
Il vincolo interrelazionale più importante è quello noto come integrità referenziale.

L'integrità referenziale è un insieme di regole che garantisce la consistenza dei dati in relazioni associate tra loro attraverso chiavi esterne; permette di controllare che non vengano eliminati o modificati per errore dati correlati.

> Date una tabella primaria e una tabella associata correlata, il vincolo di **integrità referenziale** (referential integrity constraint) sulla tabella associata è rispettato quando, per ogni valore non nullo della chiave esterna nella tabella associata, esiste un valore corrispondente della chiave primaria nella tabella primaria.

Loescher Editore - Vietata la vendita e la diffusione

FORMALMENTE

Data una relazione *R1* con chiave *K1* (*R1(K1,...)*) e una relazione *R2* con chiave esterna *KE2* associata a *K1* (*R2(...,KE2)*), il vincolo di integrità referenziale è rispettato se per ogni valore di *KE2* non nullo esiste un valore uguale di *K1* in una delle tuple della relazione *R1*.

Quando viene applicata l'integrità referenziale ci sono alcune **operazioni non consentite**:

- non è possibile inserire nel campo chiave esterna della tabella associata un valore che non esiste nella chiave primaria della tabella primaria;

  È possibile però inserire un valore nullo nella chiave esterna per rappresentare il fatto che la riga non è correlata ad alcuna riga della tabella primaria. **Nota**

- non è possibile cancellare dalla tabella primaria una riga se il valore della chiave primaria è presente nella chiave esterna di una o più righe della tabella associata;
- non è possibile modificare nella tabella primaria il valore della chiave primaria di una riga se tale valore è presente nella chiave esterna di una o più righe della tabella associata.

  Cioè non è possibile cancellare o modificare nella tabella primaria una riga che ha delle righe correlate nella tabella associata. **Nota**

Nazioni (NomeN, Continente, Capitale, Superficie, Popolazione)
Continenti (Sigla, NomeCon)
Città (NomeC, Nazione, Popolazione)

Se è definito il vincolo di integrità referenziale da *Nazioni.Continente* a *Continenti.Sigla* non è possibile inserire una nazione se non esiste la tupla relativa al continente a cui appartiene; non è possibile cancellare un continente finché esiste almeno una tupla relativa a una nazione appartenente a quel continente.

Si può scegliere di non definire l'integrità referenziale da {*Nazioni.Capitale*, *Nazioni.NomeN*} a {*Città.NomeC*, *Città.Nazione*} per permettere di inserire una nazione anche se non esiste nella relazione *Città* la tupla corrispondente alla capitale; i dati della capitale possono essere inseriti successivamente.

Se si applica l'integrità referenziale però vengono controllati anche gli eventuali errori di scrittura. **Nota**

2.2 ALGEBRA RELAZIONALE

Si possono eseguire interrogazioni sul database tramite l'algebra relazionale.

L'**algebra relazionale** utilizza operatori definiti su relazioni, che producono come risultato nuove relazioni.

Per eseguire operazioni complesse si possono costruire espressioni combinando gli operatori; il risultato è sempre una nuova relazione.

Loescher Editore - Vietata la vendita e la diffusione

Unione, intersezione e differenza

Gli operatori di unione, intersezione e differenza sono ammessi su relazioni che hanno lo stesso numero e tipo di attributi (relazioni compatibili).

Nota

Eventualmente alcuni attributi possono essere ridenominati in modo che attributi con lo stesso tipo e significato abbiano lo stesso nome in entrambe le relazioni. L'operatore di ridenominazione è $\rho_{NomeAttributo \leftarrow NuovoNome}(R)$.

Unione

$R = R1 \cup R2$

La relazione risultante contiene tutte le tuple che appartengono all'una o all'altra relazione, prendendo una sola volta quelle eventualmente uguali.

L'unione della tabella delle nazioni dell'Europa e della tabella delle nazioni dell'Asia produce la tabella delle nazioni del continente euroasiatico.

Intersezione

$R = R1 \cap R2$

La relazione risultante contiene le tuple che appartengono ad entrambe le relazioni.

L'intersezione tra la tabella delle nazioni dell'Europa e la tabella delle nazioni della NATO produce la tabella delle nazioni europee che appartengono alla NATO.

Differenza

$R = R1 - R2$

La relazione risultante contiene tutte le tuple che appartengono alla prima relazione ma non alla seconda.

La differenza tra la tabella delle nazioni dell'Europa e la tabella delle nazioni della NATO produce la tabella delle nazioni europee che non appartengono alla NATO.

Selezione

La selezione permette di scegliere le tuple di una relazione che soddisfano a una condizione.

$R = \sigma_P(R1)$

La selezione (o restrizione) di una relazione rispetto a una proprietà *P* genera una nuova relazione costituita solo dalle tuple per cui la proprietà *P* è vera.
Viene indicata mediante il simbolo σ seguito dal nome (che può essere posto tra parentesi) della relazione su cui si opera e ha come indice la proprietà che caratterizza l'operazione di selezione.
La relazione risultante ha gli stessi attributi della relazione di partenza. Le tuple della relazione sono tutte quelle che soddisfano la proprietà specificata (da 0 a tutte le tuple della relazione di partenza).

Grado e cardinalità

La relazione risultante ha lo stesso grado della relazione di partenza e cardinalità minore o al massimo uguale.

Loescher Editore - Vietata la vendita e la diffusione

$\sigma_{Superficie<100}$(Nazioni)
seleziona dalla tabella *Nazioni* tutte le nazioni con superficie minore di 100 km^2.

Proiezione

La proiezione permette di estrarre solo alcuni attributi di una relazione (eliminando poi le tuple duplicate).

$R = \Pi_L(R1)$

La proiezione di una relazione rispetto ad una lista *L* di attributi genera una nuova relazione costituita solo dagli attributi specificati. Le tuple duplicate compaiono una volta sola.

Nota
In realtà per motivi di efficienza i DBMS di solito controllano l'unicità delle tuple solo se richiesto esplicitamente dall'utente.

Viene indicata mediante il simbolo Π seguito dal nome (che può essere posto tra parentesi) della relazione su cui si opera e ha come indice l'elenco degli attributi da estrarre.

La relazione risultante ha meno attributi della relazione di partenza; ha lo stesso numero di attributi se la proiezione viene fatta allo scopo di riordinare gli attributi. Le tuple della relazione sono tutte quelle della relazione di partenza, o meno se si ottengono delle tuple uguali.

Grado e cardinalità

La relazione risultante ha grado minore della relazione di partenza, o al massimo uguale, e cardinalità uguale, o minore se risultano tuple duplicate.

$\Pi_{NomeN,\ Capitale}$(Nazioni)
estrae dalla tabella *Nazioni* il nome e la capitale di ogni nazione.

$\Pi_{Continente,\ TipoGoverno}$(Nazioni)
estrae dalla tabella *Nazioni* le colonne *Continente* e *TipoGoverno*; eliminando le righe duplicate si ottiene l'elenco dei tipi di governo presenti in ciascun continente.

Join

L'operazione di **join** (giunzione) permette di correlare dati contenuti in relazioni diverse che hanno uno o più attributi in comune o dello stesso tipo.

Nota
Per attributi comuni si intende in generale che hanno lo stesso significato; i valori degli attributi possono essere anche del tutto diversi tra di loro; in tal caso la relazione risultante non contiene tuple. Se tutti gli attributi sono comuni il join coincide con l'intersezione.

Join interno: join naturale e equi-join

$R = R1 \bowtie R2$

Il **join naturale** si può eseguire su due relazioni che hanno uno o più attributi in comune con lo stesso nome e genera una relazione che contiene tutti gli attributi delle due relazioni, con quelli comuni presi una volta sola, combinando le tuple delle due relazioni che hanno lo stesso valore negli attributi comuni, cioè concatenando ogni tupla della prima relazione con ogni tupla della seconda relazione che ha lo stesso valore negli attributi comuni.

Loescher Editore - Vietata la vendita e la diffusione

Il join naturale è indicato con il simbolo ⋈ tra i nomi delle due relazioni di partenza.

Il join naturale è commutativo e associativo.

Grado e cardinalità

Se *m* e *n* sono i gradi delle relazioni di partenza, e *k* e il numero di attributi comuni, la relazione risultante ha grado *m+n-k* (perché gli attributi comuni compaiono una volta sola). La cardinalità dipende dalla presenza o meno di valori uguali negli attributi comuni.

Se ciascuna tupla delle due relazioni di partenza contribuisce ad almeno una tupla del risultato si parla di **join completo**.

Le tuple di una relazione che non contribuiscono al risultato perché l'altra relazione non contiene tuple con lo stesso valore sull'attributo comune si chiamano **dangling** (dondolanti).

Esempio 2.1

Join naturale su un solo attributo.

Nazioni (NomeN, SiglaCont, Capitale)
Continenti (SiglaCont, NomeCont)

Nazioni ⋈ Continenti

esegue il join sul campo comune *SiglaCont* e produce una tabella con tutte le colonne di *Nazioni* più il nome del continente.

Nazioni

NomeN	SiglaCont	Capitale
Austria	EU	Vienna
Brasile	SA	Brasilia
Canada	NA	Ottawa
Egitto	AF	Il Cairo
Francia	EU	Parigi
Germania	EU	Berlino
Italia	EU	Roma
Stati Uniti	NA	Washington

Continenti

SiglaCont	NomeCont
AS	Asia
EU	Europa
NA	Nord America

Nota

Questa situazione è possibile solo se non è stabilito nessun vincolo di integrità referenziale sulle relazioni.

Nazioni ⋈ Continenti

NomeN	SiglaCont	Capitale	NomeCont
Austria	EU	Vienna	Europa
Canada	NA	Ottawa	Nord America
Francia	EU	Parigi	Europa
Germania	EU	Berlino	Europa
Italia	EU	Roma	Europa
Stati Uniti	NA	Washington	Nord America

Le tuple di *Nazioni* relative a Brasile ed Egitto con *SiglaCont* 'SA' e 'AF' e la tupla di *Continenti* con sigla 'AS' sono dangling.

Esempio 2.2

Join naturale su più attributi.

Città (IdCittà, NomeC, NomeNazione)
Nazioni (NomeNazione, Continente, IdCittà), dove *IdCittà* identifica la capitale.

Loescher Editore - Vietata la vendita e la diffusione

Nazioni ⋈ Città

esegue il join sui campi comuni *NomeNazione* e *IdCittà* e produce una tabella con tutte le colonne di *Nazioni* più la colonna *NomeC*.
Le righe contenute nella nuova tabella sono solo quelle con valore uguale nei due campi comuni, perciò non compaiono nella tabella le città che non sono capitali e le eventuali nazioni per cui non esiste una riga corrispondente nella tabella *Città* (ciò può avvenire solo se non è definito il vincolo di integrità referenziale).

Se l'attributo comune non ha lo stesso nome, si deve indicare la condizione di uguaglianza sui due attributi da correlare come indice del simbolo ⋈ e in tal caso si parla di **equi-join**.

Esempio 2.3

Equi-join.

Nazioni (Nome, Continente, Capitale)
Continenti (Sigla, Nome)

Nazioni $\bowtie_{\text{Continente=Sigla}}$ Continenti

esegue il join sui campi *Continente* e *Sigla* e decodifica il nome del continente.

Nazioni

Nome	Continente	Capitale
Austria	EU	Vienna
Brasile	SA	Brasilia
Canada	NA	Ottawa
Egitto	AF	Il Cairo
Francia	EU	Parigi
Germania	EU	Berlino
Italia	EU	Roma
Stati Uniti	NA	Washington

Continenti

Sigla	Nome
AS	Asia
EU	Europa
NA	Nord America

Nota

Questa situazione è possibile solo se non è stabilito nessun vincolo di integrità referenziale sulle relazioni.

Nazioni $\bowtie_{\text{Continente=Sigla}}$ Continenti

Nome	Continente	Capitale	Sigla	Nome
Austria	EU	Vienna	EU	Europa
Canada	NA	Ottawa	NA	Nord America
Francia	EU	Parigi	EU	Europa
Germania	EU	Berlino	EU	Europa
Italia	EU	Roma	EU	Europa
Stati Uniti	NA	Washington	NA	Nord America

Con il join naturale e l'equi-join, se una tupla di una relazione non ha una tupla corrispondente nell'altra relazione, non compare nel risultato. Il join naturale e l'equi-join pertanto sono anche indicati come **join interno** (**inner join**).

Join esterno

Se si desidera che nel risultato compaiano tutte le tuple di una relazione anche se non ci sono tuple corrispondenti nell'altra, si può utilizzare il join esterno.

Loescher Editore - Vietata la vendita e la diffusione

Con il **join esterno** (**outer join**) nel risultato vengono considerate tutte le tuple, eventualmente completandole con valori *null*.
Ci sono tre varianti di join esterno:

- il join esterno sinistro (**left join**) estende solo le tuple del primo operando; nel risultato compaiono tutte le tuple della prima relazione e solo quelle della seconda che hanno un valore corrispondente;
- il join esterno destro (**right join**) estende solo le tuple del secondo operando; nel risultato compaiono tutte le tuple della seconda relazione e solo quelle della prima che hanno un valore corrispondente;
- il join esterno completo (**full join**) estende tutte le tuple; nel risultato compaiono tutte le tuple sia della prima che della seconda relazione, anche se non hanno valori corrispondenti.

Esempio 2.4

Join esterni.

Date le relazioni:

Nazioni (NomeN, SiglaCont, Capitale)
Continenti (SiglaCont, NomeCont)

Nazioni

NomeN	SiglaCont	Capitale
Austria	EU	Vienna
Brasile	SA	Brasilia
Canada	NA	Ottawa
Egitto	AF	Il Cairo
Francia	EU	Parigi
Germania	EU	Berlino
Italia	EU	Roma
Stati Uniti	NA	Washington

Continenti

SiglaCont	NomeCont
AS	Asia
EU	Europa
NA	Nord America

Nota

Questa situazione è possibile solo se non è stabilito nessun vincolo di integrità referenziale sulle relazioni.

si possono eseguire i seguenti join esterni:

- join sinistro;

  Nazioni $\bowtie_{LEFT}$ **Continenti**

  Le righe contenute nella nuova tabella comprendono anche le eventuali nazioni per cui non esiste una riga corrispondente nella tabella *Continenti*.

Nazioni $\bowtie_{LEFT}$ **Continenti**

NomeN	SiglaCont	Capitale	SiglaCont	NomeCont
Austria	EU	Vienna	EU	Europa
Brasile	SA	Brasilia	NULL	NULL
Canada	NA	Ottawa	NA	Nord America
Egitto	AF	Il Cairo	NULL	NULL
Francia	EU	Parigi	EU	Europa
Germania	EU	Berlino	EU	Europa
Italia	EU	Roma	EU	Europa
Stati Uniti	NA	Washington	NA	Nord America

Loescher Editore - Vietata la vendita e la diffusione

- join destro;

  Nazioni $\bowtie_{RIGHT}$ Continenti

  Le righe contenute nella nuova tabella comprendono tutti i continenti anche quelli per cui non sono state inserite nazioni.

  Nazioni $\bowtie_{RIGHT}$ Continenti

NomeN	SiglaCont	Capitale	SiglaCont	NomeCont
Austria	EU	Vienna	EU	Europa
Canada	NA	Ottawa	NA	Nord America
Francia	EU	Parigi	EU	Europa
Germania	EU	Berlino	EU	Europa
Italia	EU	Roma	EU	Europa
Stati Uniti	NA	Washington	NA	Nord America
NULL	NULL	NULL	AS	Asia

- join completo;

  Nazioni $\bowtie_{FULL}$ Continenti

  Nella tabella risultante compaiono tutte le nazioni e tutti i continenti.

  Nazioni $\bowtie_{FULL}$ Continenti

NomeN	SiglaCont	Capitale	SiglaCont	NomeCont
Austria	EU	Vienna	EU	Europa
Brasile	SA	Brasilia	NULL	NULL
Canada	NA	Ottawa	NA	Nord America
Egitto	AF	Il Cairo	NULL	NULL
Francia	EU	Parigi	EU	Europa
Germania	EU	Berlino	EU	Europa
Italia	EU	Roma	EU	Europa
Stati Uniti	NA	Washington	NA	Nord America
NULL	NULL	NULL	AS	Asia

Self-join

Con il **self-join** le tuple di una relazione vengono combinate con le tuple della stessa relazione nel caso di attributi con valori corrispondenti, cioè con lo stesso dominio.
Di solito si stabilisce l'integrità referenziale tra gli attributi corrispondenti.

Esempio 2.5

Self-join.

Fiumi (NomeF, Lunghezza, AffluenteDi)
AffluenteDi è chiave esterna, in integrità referenziale con *NomeF*.

Fiumi $\bowtie_{Self}$ Fiumi

o più formalmente Fiumi $\bowtie_{AffluenteDi=NomeF2}$ $\rho_{NomeF2,Lunghezza2,AffluenteDi2 \leftarrow NomeF,Lunghezza,AffluenteDi}$(Fiumi)

La tabella *Fiumi* viene congiunta con una copia di se stessa sul campo *AffluenteDi* della prima copia e *NomeF* della seconda.
Permette di ottenere le righe corrispondenti ai fiumi che sono affluenti di altri fiumi.

Loescher Editore - Vietata la vendita e la diffusione

Theta-join

La forma più generale di join è il **theta-join** che può essere definito come un prodotto cartesiano seguito da una selezione.

Esempio 2.6

Theta-join.

FiumiEuropa (NomeF, Lunghezza)
FiumiAmerica (NomeF, Lunghezza)

FiumiEuropa $\bowtie_{\text{FiumiEuropa.Lunghezza>FiumiAmerica.Lunghezza}}$ FiumiAmerica

esegue il prodotto cartesiano, poi seleziona le righe in cui i fiumi europei sono più lunghi dei fiumi americani.

Il **prodotto cartesiano** di due relazioni è la relazione in cui ogni tupla è ottenuta concatenando ogni tupla della prima relazione con ogni tupla della seconda.

Il prodotto cartesiano di due relazioni R e S si indica con RxS.

Prodotto cartesiano

$RxS = \{(r_1, r_2, \dots r_n, s_1, s_2, \dots s_m) \mid (r_1, r_2, \dots r_n) \in R, (s_1, s_2, \dots s_m) \in S\}$

Il prodotto cartesiano ha per grado la somma dei gradi delle singole relazioni ($n+m$) e per cardinalità il prodotto delle cardinalità (nxm).

Tutte le forme di join sono casi particolari di theta-join, infatti se la selezione è per una condizione di uguaglianza tra un attributo della prima relazione e uno della seconda si ha un equi-join; se l'attributo ha lo stesso nome si ha un join naturale.
Il join naturale di due relazioni può essere definito come il prodotto cartesiano delle due relazioni seguito dalla selezione delle tuple che hanno lo stesso valore per i due attributi con lo stesso nome e da una proiezione per eliminare uno degli attributi comuni.

Nota

In pratica i DBMS non usano i nomi di attributo per correlare le relazioni e quindi non usano join naturali ma equi-join e theta join.

Esempi di espressioni in algebra relazionale

Gli operatori possono essere combinati per ottenere espressioni complesse.

Negli esempi successivi si fa riferimento al seguente schema relazionale.

Nota

La progettazione del modello concettuale e la derivazione dello schema relazionale relativo sono riportate negli esempi 4.38 e 5.20 (capitoli 4 e 5).

Nazioni (NomeN, Continente, Capitale, Superficie, Popolazione)
Continenti (Sigla, NomeCon)
Città (IdCittà, NomeC, Nazione, Popolazione)
Mari (NomeM, Oceano) in cui Oceano è un valore booleano
Fiumi (NomeF, Lunghezza)
Litorale (Nazione, Mare)
Attraversamento (Nazione, Fiume)

Loescher Editore - Vietata la vendita e la diffusione

Nazioni.Continente è in integrità referenziale con *Continenti.Sigla*;
Nazioni.Capitale è in integrità referenziale con *Città.IdCittà*;
Città.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Litorale.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Litorale.Mare è in integrità referenziale con *Mari.NomeM*;
Attraversamento.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Attraversamento.Fiume è in integrità referenziale con *Fiumi.NomeF*.

Esempio 2.7

Nome, popolazione, capitale e popolazione della capitale di tutte le nazioni dell'Europa.

Le relazioni interessate sono:
Nazioni (NomeN, Continente, Capitale, Superficie, Popolazione)
Continenti (Sigla, NomeCon)
Città (IdCittà, NomeC, Nazione, Popolazione)

$\Pi_{\text{NomeN, Nazioni.Popolazione, NomeC, Città.Popolazione}}((\sigma_{\text{NomeCon='Europa'}}(\text{Nazioni} \bowtie_{\text{Continente=Sigla}} \text{Continenti})) \bowtie_{\text{Capitale=IdCittà}} \text{Città})$

Temp1 = Nazioni $\bowtie_{\text{Continente=Sigla}}$ Continenti
join di *Nazioni* e *Continenti*;

Temp2 = $\sigma_{\text{NomeCon='Europa'}}$ (Temp1)
selezione sul risultato precedente delle righe in cui *NomeCon* è 'Europa';

Temp3 = Temp2 $\bowtie_{\text{Capitale=IdCittà}}$ Città
join del risultato precedente con *Città*;

$\Pi_{\text{NomeN, Nazioni.Popolazione, NomeC, Città.Popolazione}}$ (Temp3)
proiezione sul risultato precedente di *NomeN*, *Nazioni.Popolazione*, *NomeC* e *Città.Popolazione*.

Esempio 2.8

Nome e superficie delle nazioni che sono attraversate dal Danubio.

Le relazioni interessate sono:
Nazioni (NomeN, Continente, Capitale, Superficie, Popolazione)
Attraversamento (Nazione, Fiume)

$\Pi_{\text{NomeN, Superficie}}((\sigma_{\text{Fiume='Danubio'}}(\text{Attraversamento})) \bowtie_{\text{Attraversamento.Nazione=Nazioni.NomeN}} \text{Nazioni})$

Temp1 = $\sigma_{\text{Fiume='Danubio'}}$ (Attraversamento)
selezione delle righe di *Attraversamento* in cui *Fiume* è 'Danubio';

Temp2 = Temp1 $\bowtie_{\text{Attraversamento.Nazione=Nazioni.NomeN}}$ Nazioni
join del risultato precedente con *Nazioni*;

$\Pi_{\text{NomeN, Superficie}}$ (Temp2)
proiezioni sul risultato precedente dei campi *Nome* e *Superficie*.

Esempio 2.9

Nome delle nazioni dell'Europa che sono bagnate dal mare.

Loescher Editore - Vietata la vendita e la diffusione

Le relazioni interessate sono:
Nazioni (NomeN, Continente, Capitale, Superficie, Popolazione)
Continenti (Sigla, NomeCon)
Litorale (Nazione, Mare)

$(\Pi_{NomeN} (\sigma_{NomeCon='Europa'} (Nazioni \bowtie_{Continente=Sigla} Continenti))) \cap (\Pi_{Nazione} (Litorale))$

Temp1 = Nazioni $\bowtie_{Continente=Sigla}$ Continenti
join di *Nazioni* e *Continenti*;

Temp2 = $\sigma_{NomeCon='Europa'}$ (Temp1)
selezione sul risultato precedente delle righe in cui *NomeCon* è "Europa";

Temp3 = Π_{NomeN} (Temp2)
proiezione sul risultato precedente di *NomeN* (si ottiene il nome delle nazioni europee);

Temp4 = $\Pi_{Nazione}$ (Litorale)
proiezione su *Litorale* di *NomeN* (si ottiene il nome di tutte le nazioni bagnate dal mare;

Temp3 $\cap$ Temp4
Intersezione dei due risultati parziali (entrambi costituiti da un'unica colonna con i nomi delle nazioni, una delle quali deve essere ridenominata perchè siano omonime).

ESEMPIO 2.10

Nome delle nazioni dell'Europa che non sono bagnate dal mare.

Basta modificare l'operazione di intersezione con la differenza.

Temp3 – Temp4
Differenza dei due risultati parziali (entrambi costituiti da un'unica colonna con i nomi delle nazioni, una delle quali deve essere ridenominata perchè siano omonime).

2.3 VISTE

Le viste permettono di ottenere rappresentazioni diverse degli stessi dati.

Le **viste** vengono definite per mezzo di espressioni del linguaggio di interrogazione, che possono riferirsi anche ad altre viste.

Le interrogazioni effettuate su una vista che si basa su un'altra vista vengono risolte sostituendo alla vista di base la sua definizione, componendo le interrogazioni.

Le viste possono essere utili per diversi motivi:

- per definire espressioni complesse o ripetute;
- per considerare solo una porzione del database eliminando ciò che non è rilevante per l'applicazione, eventualmente definendo anche autorizzazioni di accesso sulle viste;
- per mantenere la compatibilità con relazioni non più esistenti in caso di ristrutturazioni del database.

Nota

Ci sono limitazioni alla possibilità di specificare aggiornamenti operando sulle viste.

Le **viste virtuali** devono essere ricalcolate ogni volta.
Un'alternativa sono le **viste materializzate** che vengono memorizzate nel database; i dati delle

Loescher Editore - Vietata la vendita e la diffusione

viste materializzate sono immediatamente disponibili, ma devono essere aggiornati in modo coerente con il database.
Le viste materializzate sono convenienti quando gli aggiornamenti sono rari rispetto alle interrogazioni e il calcolo della vista è complesso.

Nota

I DBMS attuali permettono solo l'uso di viste virtuali.

VERIFICA

Questionario

1. Che cos'è una relazione?
2. Che cosa sono gli attributi di una relazione?
3. Che cosa sono le tuple?
4. Che cosa si intende per schema relazionale?
5. Che cos'è una istanza di una relazione?
6. Che cos'è il grado di una relazione?
7. Che cos'è la cardinalità di una relazione?
8. Che cos'è una chiave candidata e quali proprietà deve soddisfare?
9. Che cos'è una superchiave?
10. Che cos'è la chiave primaria?
11. Che cos'è una chiave esterna?
12. Che cosa sono i vincoli di integrità?
13. Qual è la differenza tra vincoli intrarelazionali e vincoli interrelazionali?
14. Che cos'è l'integrità referenziale?
15. A che cosa serve l'algebra relazionale?
16. In che cosa consistono le operazioni di unione, intersezione e differenza?
17. In che cosa consiste l'operazione di selezione?
18. In che cosa consiste l'operazione di proiezione?
19. In che cosa consiste l'operazione di join?
20. Qual è la differenza tra join naturale e equi-join?
21. Quali sono le diverse forme di join esterno?
22. Che cos'è il self-join?
23. Che cos'è il theta-join?
24. Che cos'è una vista?

Test

1 ..... *Indicare se le seguenti affermazioni sono vere o false.*

V	F	
☐	☐	1) L'istanza di una relazione è l'insieme delle tuple della relazione.
☐	☐	2) L'istanza di una relazione è variabile nel tempo.
☐	☐	3) L'istanza è il significato intensionale di una relazione.
☐	☐	4) L'istanza è il significato estensionale di una relazione.

Loescher Editore - Vietata la vendita e la diffusione

V	F	
☐	☐	5) In due o più relazioni possono essere presenti attributi con lo stesso nome.
☐	☐	6) La chiave primaria di una relazione può contenere valori *null*.
☐	☐	7) Una chiave candidata che non è stata scelta come chiave primaria può contenere valori *null*.
☐	☐	8) Ogni relazione ha sempre almeno una chiave candidata.
☐	☐	9) Gli attributi di una superchiave hanno sempre valori diversi tra loro in tutte le tuple.
☐	☐	10) Possono esserci attributi che non appartengono a nessuna chiave candidata.
☐	☐	11) Ogni attributo appartiene al massimo a una chiave candidata.
☐	☐	12) La chiave può essere un valore generato automaticamente dal DBMS.
☐	☐	13) Una chiave esterna può essere formata da più attributi.
☐	☐	14) Un vincolo di chiave è un vincolo intrarelazionale.
☐	☐	15) Un vincolo di dominio si riferisce a più valori.
☐	☐	16) Un vincolo su valori è un vincolo interrelazionale.
☐	☐	17) Un vincolo di integrità referenziale associa la chiave primaria di due relazioni.
☐	☐	18) L'operazione di unione è sempre possibile su qualsiasi coppia di relazioni.
☐	☐	19) L'operazione di selezione è un operatore unario cioè agisce su una sola relazione.
☐	☐	20) L'operazione di proiezione genera una relazione con numero di attributi minore o uguale rispetto alla relazione di partenza.
☐	☐	21) Il risultato di un join contiene sempre tutte le tuple delle due relazioni su cui agisce.
☐	☐	22) L'operazione di join può sempre essere vista come un prodotto cartesiano seguito da una selezione.

2 ..... *Data la relazione R(A,B,C,D,E) indicare quali delle seguenti proiezioni hanno certamente lo stesso numero di tuple di R:*

- ☐ $\Pi_{A,B,C,D}(R)$
- ☐ $\Pi_{A,C}(R)$
- ☐ $\Pi_{B,C}(R)$
- ☐ $\Pi_{B,C,D,E}(R)$
- ☐ $\Pi_{C}(R)$
- ☐ $\Pi_{C,D}$

3 ..... *Individuare la risposta esatta.*

1) Date le relazioni R1(A,B,C,D) e R2(B,C,E) il join naturale R1⋈R2 ha lo schema
   - ☐ R(A,B,C,B,C,D,E);
   - ☐ R(A,B,C,D,E);
   - ☐ R(A,D,E).

2) Date le relazioni R1(A,B,C) e R2(B,D,E) dove R1.B è in integrità referenziale con R2.B, il join naturale R1⋈R2 ha:
   - ☐ esattamente la stessa cardinalità del join esterno sinistro;
   - ☐ esattamente la stessa cardinalità del join esterno destro;
   - ☐ esattamente la stessa cardinalità del join esterno completo.

Loescher Editore - Vietata la vendita e la diffusione

3) Date le relazioni R1(A,B,C) e R2(B,D,E) senza nessun vincolo di integrità referenziale riguardo ai join esterni tra R1 E R2 si può dire che:

- ☐ il join sinistro ha cardinalità maggiore o uguale del join destro;
- ☐ il join destro ha cardinalità maggiore o uguale del join sinistro;
- ☐ il join completo ha cardinalità maggiore o uguale sia del join sinistro che del join destro.

Esercizi

1 ..... Date le tabelle:

Opere

Titolo	Poesia	Autore	Data
Promessi Sposi	F	Manzoni	1840
Il fu Mattia Pascal	F	Pirandello	1904
L'infinito	V	Leopardi	NULL
Rosso Malpelo	F	Verga	NULL
La pioggia nel pineto	V	D'Annunzio	NULL
Il passero solitario	V	Leopardi	NULL
I Malavoglia	F	Verga	1881
Il cinque maggio	V	Manzoni	NULL
Uno, nessuno, centomila	F	Pirandello	1925
...	...	...	...

Scrittori

Cognome	Nome	AnnoNascita	LuogoNascita	AnnoMorte	LuogoMorte
Carducci	Giosuè	1835	Val di Castello	1907	Bologna
Manzoni	Alessandro	1785	Milano	1873	NULL
Leopardi	Giacomo	1798	Recanati	1837	Napoli
Pascoli	Giovanni	1855	San Mauro	1912	Bologna
Verga	Giovanni	1840	Catania	1922	NULL
...	...	...	...	...	...

- di ciascuna relazione rappresentare lo schema, individuando anche il formato degli attributi;
- per ciascuna relazione individuare le chiavi candidate e fissare la chiave primaria;
- per ciascuna relazione individuare i vincoli di integrità intrarelazionali;
- individuare i vincoli di integrità referenziale (in base ai dati presenti, al momento dell'inserimento erano stabiliti vincoli di integrità referenziale? quali sono i dati che non potrebbero essere presenti in base a tale vincolo?);
- determinare mediante operazioni dell'algebra relazionale:
  1) le opere di Pirandello;
  2) gli scrittori nati nella prima metà dell'ottocento;
  3) cognome, nome, anno di nascita e di morte degli scrittori;
  4) i titoli delle poesie;
  5) i titoli di tutte le opere con nome e cognome del loro autore;
  6) i titoli delle sole poesie con nome e cognome del loro autore;
- in base ai dati presenti nelle tabelle costruire la tabella risultante dalle operazioni:
  1) join esterno sinistro tra *Opere* e *Scrittori*;
  2) join esterno destro tra *Opere* e *Scrittori*;
  3) join esterno completo tra *Opere* e *Scrittori*.

Loescher Editore - Vietata la vendita e la diffusione

2 ..... Dato lo schema relazionale
Nazioni (NomeN, Continente, Capitale, Superficie, Popolazione)
Continenti (Sigla, NomeCon)
Città (IdCittà, NomeC, Nazione, Popolazione)
Mari (NomeM, Oceano) in cui Oceano è un valore booleano
Fiumi (NomeF, Lunghezza)
Litorale (Nazione, Mare)
Attraversamento (Nazione, Fiume)
con i vincoli di integrità referenziale specificati a pagina 30, determinare mediante operazioni dell'algebra relazionale:
1) nome e capitale di tutte le nazioni asiatiche;
2) nome e popolazione di tutte le città italiane;
3) nome di tutti i mari che bagnano le nazioni europee;
4) nome e lunghezza di tutti i fiumi americani;
5) nome e continente delle nazioni che non sono bagnate da nessun mare;
6) nome e continente delle nazioni che non sono attraversate da nessun fiume;
7) nome di tutte le nazioni bagnate dall'oceano Pacifico;
8) nome di tutti i mari che bagnano l'Europa;
9) nome degli oceani che bagnano l'Africa.

3 ..... Dato lo schema relazionale
Studenti (Matricola, Cognome, Nome)
Materie (CodMateria, NomeMateria)
Esami (Studente, Materia, Voto, Data)
Esami.Studente è in integrità referenziale con *Studenti.Matricola*;
Esami.Materia è in integrità referenziale con *Materie.CodMateria*;
determinare mediante operazioni dell'algebra relazionale:
1) per ogni studente, cognome, nome e elenco degli esami sostenuti con materia, voto e data;
2) nome degli studenti che hanno preso 30 in almeno un esame;
3) nome degli studenti che hanno preso 30 nell'esame "Analisi1";
4) elenco delle materie d'esame;
5) matricola e nome degli studenti che hanno sostenuto esami nell'anno corrente.

4 ..... Dato lo schema relazionale
Persona (CodiceFiscale, Nome, Cognome, AnnoNascita, Sesso)
Matrimonio (Marito, Moglie, AnnoMatrimonio, LuogoMatrimonio)
Matrimonio.Marito è in integrità referenziale con *Persona.CodiceFiscale*;
Matrimonio.Moglie è in integrità referenziale con *Persona.CodiceFiscale*;
determinare mediante operazioni dell'algebra relazionale:
1) per ogni coppia, nome, cognome e anno di nascita del marito e della moglie e luogo e anno del matrimonio;
2) nome e cognome delle persone che si sono sposate entro i 20 anni di età;
3) nome e cognome di marito e moglie che sono nati nello stesso posto;
4) nome e cognome di marito e moglie che sono nati nello stesso posto e si sono sposati in quello stesso posto.

5 ..... Dato lo schema relazionale
Fiumi (NomeF, Lunghezza, AffluenteDi)
AffluenteDi è in integrità referenziale con *NomeF*;
determinare mediante operazioni dell'algebra relazionale:
1) i fiumi che sono affluenti di un fiume più corto;
2) tutti gli affluenti diretti del fiume Po.

Loescher Editore - Vietata la vendita e la diffusione

3 SQL

Prerequisiti

- Sapere applicare i concetti del modello relazionale
- Saper effettuare interrogazioni con l'algebra relazionale

Obiettivi

Conoscenze

- Caratteristiche generali del linguaggio: evoluzione degli standard, identificatori e tipi di dati, significato dei valori nulli
- Comandi per la definizione di elementi dello schema
- Comandi per l'inserimento, la cancellazione e l'aggiornamento di tuple
- Comando SELECT: clausole FROM, WHERE, GROUP BY, HAVING, ORDER BY, funzioni di aggregazione, operazioni di join, sottoquery
- Trigger
- Stored procedure
- Comandi GRANT e REVOKE e ruoli
- Transazioni e proprietà ACID, comandi COMMIT e ROLLBACK

Abilità/capacità

- Definire lo schema
- Popolare il database, modificare e cancellare record
- Effettuare interrogazioni che coinvolgono una o più tabelle

Loescher Editore - Vietata la vendita e la diffusione

3.1 IL LINGUAGGIO SQL

SQL (Structured Query Language) è un linguaggio dichiarativo che è diventato lo standard per la gestione dei database relazionali.

SQL ha subìto un processo di standardizzazione ad opera di ANSI ed ISO che ha avuto inizio nella prima metà degli anni ottanta e continua ancora oggi; partecipano alla definizione degli standard anche gran parte dei produttori del settore.

I diversi DBMS offrono **implementazioni** diverse di SQL.

Normalmente è implementato SQL-2, ma ci sono diversi livelli di aderenza allo standard che vengono quantificati come Entry SQL (simile a SQL-89), Intermediate SQL e Full SQL. **Nota**

SQL permette di gestire in modo standard database relazionali, sia per quanto riguarda le interrogazioni che le funzioni di gestione relative a DLL e DML.
Spesso i DBMS mettono a disposizione interfacce grafiche che permettono di indicare in modo semplice i risultati desiderati e generano automaticamente le istruzioni SQL corrispondenti.
SQL può anche essere usato in applicazioni realizzate con linguaggi di programmazione tradizionali (linguaggi ospite) e in applicazioni server side.

STANDARD RELATIVI A SQL

La prima definizione dello standard è stata emanata nel 1986 dall'ANSI (**SQL-86**), seguita da uno standard ISO nel 1987, ed estesa negli anni seguenti (**SQL-89**).
Nel 1992 è stata pubblicata una nuova versione dello standard (ISO 9075), noto come **SQL-2**) con molte nuove funzionalità.
La versione successiva, (ISO 9075-2, nota come **SQL-3**), è un'estensione molto significativa, pienamente compatibile con SQL-2; lo standard SQL-3 è organizzato in diverse parti numerate e denominate, ciascuna dedicata a un particolare aspetto del linguaggio (per esempio la parte 13 SQL/JRT è relativa all'integrazione con Java e la parte 14 SQL/XML è relativa alle interazioni tra SQL e XML).
La versione di SQL-3 indicata come **SQL:1999** include nuovi servizi come i trigger, le viste ricorsive e il supporto per il paradigma a oggetti.
La versione **SQL:2003** modifica ulteriormente lo standard introducendo nuove parti, ma anche eliminando alcuni costrutti degli standard precedenti che non erano mai stati implementati e ritenuti obsoleti.

Tabella 3.1 *Standardizzazioni di SQL*

Nome informale	Nome ufficiale	Caratteristiche
SQL base	SQL-86	Costrutti base
	SQL-89	Integrità referenziale
SQL-2	SQL-92	Vari costrutti nuovi
		Tre livelli di implementazione: Entry, Intermediate e Full
SQL-3	SQL:1999	Organizzato in diverse parti
		Nuovi servizi: Trigger, viste ricorsive, supporto per il paradigma a oggetti
	SQL:2003	Estensioni del modello a oggetti
		Eliminazione di costrutti non usati
		Nuove parti: SQL/JRT, SQL/XML ...

Loescher Editore - Vietata la vendita e la diffusione

3.2 PRIMI ELEMENTI

Le istruzioni SQL possono essere scritte indifferentemente in maiuscolo o minuscolo.

Nota: Le istruzioni risultano più leggibili se le parole riservate sono scritte in maiuscolo.

In molte implementazioni ogni istruzione termina con un punto e virgola.

Formato dei comandi

La sintassi delle istruzioni viene descritta con il seguente formato:

- parentesi quadre [ ] per indicare valori opzionali;
- il simbolo | per indicare alternative tra più valori;
- parentesi graffe { } per racchiudere valori che sono in alternativa tra loro, ma di cui uno deve essere specificato obbligatoriamente.

Nota: I nomi di tabelle e attributi nella definizione dei comandi sono indicati in corsivo.

Gli identificatori (nomi di tabelle e attributi) devono:

- avere lunghezza massima 18 caratteri;
- iniziare con una lettera;
- contenere solo lettere, numeri e il trattino di sottolineatura (_).

Il nome degli attributi è identificato con la notazione puntata *NomeTabella.NomeAttributo*; il nome della tabella può essere omesso se non ci sono ambiguità nell'identificazione dell'attributo (è necessario solo se ci sono attributi con lo stesso nome in tabelle diverse).

Tabella 3.2 *Tipi di dati standard del linguaggio SQL*

Tipo	Descrizione	Valori
BOOLEAN	valore logico	True, False
CHARACTER(n)	stringa di lunghezza *n*	*n* da 1 a 15000
INTEGER(p)	numero intero con *p* cifre	*p* da 1 a 45
SMALLINT	numero intero con 5 cifre	da -32768 a 32767
INTEGER	numero intero con 10 cifre	da -2.147.483.648 a 2.147.483.647
DECIMAL(p,s)	numero decimale con *p* cifre di cui *s* decimali	*p* da 1 a 45 e *s* da 0 a *p*
REAL	numero decimale con 7 cifre decimali	valore assoluto da 1E-38 a 1E+38
FLOAT	numero decimale con 15 cifre decimali	valore assoluto da 1E-38 a 1E+38
FLOAT(p)	numero decimale con *p* cifre decimali	*p* da 1 a 45
DATE	data nella forma *MM/GG/AA*	
TIME	ora nella forma *HH:MM*	

Abbreviazioni

DECIMAL può essere abbreviato in DEC (o al posto di DECIMAL si può usare NUMERIC);
CHARACTER può essere abbreviato in CHAR;
INTEGER può essere abbreviato in INT;
DECIMAL equivale a DECIMAL(15,0);
CHARACTER equivale a CHARACTER(1).

Loescher Editore - Vietata la vendita e la diffusione

Le costanti stringa sono delimitate da ' o ".
I confronti tra stringhe sono fatti in base al codice ASCII.
I numeri reali hanno il punto come separatore delle cifre decimali e sono espressi in notazione esponenziale.

Nelle espressioni si possono usare gli operatori:

- aritmetici: + - * /
- relazionali: = <> < > <= >=
- logici: AND OR NOT

Il valore NULL

Gli attributi con valore non precisato hanno valore NULL.
NULL non è mai uguale a nessun altro valore, neanche allo 0 per i dati numerici o alla stringa vuota per quelli alfanumerici.
I confronti che riguardano attributi con valore NULL sono considerati sconosciuti (NULL non è mai uguale maggiore o minore di nessun altro valore). Si può solo controllare se un valore è NULL.
Negli ordinamenti NULL di solito è posto all'inizio di sequenze crescenti.

Definizione di nuovi tipi di dato

Si possono definire nuovi tipi di dato con l'istruzione CREATE DOMAIN:

CREATE DOMAIN *NomeDominio* AS *Tipo*
[CHECK (*Condizione*)];

assegna a *NomeDominio* tutti i valori di *Tipo* che verificano la condizione, se presente.

```
CREATE DOMAIN Positivo AS Integer
CHECK (VALUE>0);
```

```
CREATE DOMAIN mesi AS char(10)
CHECK (VALUE IN ('Gennaio', 'Febbraio', ..., 'Dicembre'));
```

Nota

Si può eliminare un dominio con il comando DROP DOMAIN o modificarlo con il comando ALTER DOMAIN.

Schema relazionale di riferimento

Negli esempi successivi si fa riferimento al seguente schema relazionale.

Nazioni (NomeN, Continente, Capitale, Superficie, Popolazione)
Continenti (Sigla, NomeCon)
Città (IdCittà, NomeC, Nazione, Popolazione)
Mari (NomeM, Oceano)
Fiumi (NomeF, Lunghezza)
Laghi (NomeL, Superficie)
Litorale (Nazione, Mare)
Attraversamento (Nazione, Fiume)
Presenza (Nazione, Lago)
Affluenza (Affluente, Fiume)

Loescher Editore - Vietata la vendita e la diffusione

Nazioni.Continente è in integrità referenziale con *Continenti.Sigla*;
Nazioni.Capitale è in integrità referenziale con *Città.IdCittà*;
Città.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Litorale.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Litorale.Mare è in integrità referenziale con *Mari.NomeM*;
Attraversamento.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Attraversamento.Fiume è in integrità referenziale con *Fiumi.NomeF*;
Presenza.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Presenza.Lago è in integrità referenziale con *Laghi.NomeL*;
Affluenza.Affluente è in integrità referenziale con *Fiumi.NomeF*;
Affluenza.Fiume è in integrità referenziale con *Fiumi.NomeF*.

Nota La progettazione del modello concettuale e la derivazione dello schema relazionale relativo sono riportate negli esempi 4.38 e 5.20 (capitoli 4 e 5).

3.3 OPERAZIONI DI DEFINIZIONE DEI DATI

Le operazioni definizione dei dati (DDL) sono quelle relative alla creazione della struttura del database: permettono di definire e modificare la struttura di ciascuna tabella, o di cancellare un'intera tabella.

Nota In genere nei DBMS la creazione e modifica di tabelle avviene attraverso una interfaccia grafica che permette di effettuare le operazioni in modo semplice, senza conoscere i comandi SQL.

Creazione di tabelle

Per creare una tabella si usa il comando CREATE TABLE seguito dal nome della tabella e dall'elenco degli attributi.
Per ogni attributo si deve specificare il nome e il tipo; si può specificare un valore di default da assegnare se al momento della creazione del record non viene specificato alcun valore (come valore di default si può specificare anche NULL).

CREATE TABLE *Tabella*
(*Attributo1 Tipo* [DEFAULT *ValoreDiDefault*]
, *Attributo2 ...*
...
, *AttributoN ...*
);

Nota Al posto di *Tipo* si può indicare un *Dominio* definito precedentemente.

Nella creazione della tabella si possono aggiungere dei vincoli.

CREATE TABLE *Tabella*
(*Attributo1 Tipo* [DEFAULT *ValoreDiDefault*] [***Vincoli***]
, *Attributo2 ...*
...
, *AttributoN ...*
[***Vincoli***]
);

Loescher Editore - Vietata la vendita e la diffusione

Vincoli intrarelazionali predefiniti

PRIMARY KEY identifica la chiave primaria; se la chiave è costituita da più attributi la clausola deve essere inserita dopo la dichiarazione degli attributi nella forma:

PRIMARY KEY (*Attributo1*, ... , *AttributoN*)

la clausola può essere usata in questa forma anche se la chiave è composta da un solo attributo;

NOT NULL indica che non è ammesso il valore NULL, cioè l'attributo è obbligatorio;

UNIQUE vieta la presenza di valori duplicati in una colonna o un insieme di colonne; per specificare che non deve esserci un valore ripetuto in più colonne la clausola deve essere inserita come riga isolata dopo la dichiarazione degli attributi;

Nota Un insieme di attributi col vincolo UNIQUE è una chiave candidata non primaria.

VALUE BETWEEN ... **AND** ...
vincolo di dominio che specifica un intervallo di valori ammessi;

VALUE IN (...) vincolo di dominio che specifica un elenco di valori ammessi (dominio enumerativo);

CHECK (*Condizione*)
permette di specificare un generico vincolo intrarelazionale; deve essere inserita dopo la dichiarazione degli attributi.

Vincolo interrelazionale di integrità referenziale

REFERENCES *Tabella* (*Attributo*)
specifica che un campo è chiave esterna e indica con quale campo di quale tabella è in integrità referenziale; se la chiave esterna è costituita da più campi la clausola deve essere inserita dopo la dichiarazione degli attributi nella forma:

FOREIGN KEY (*Attributo1*, ... , *AttributoN*)
REFERENCES *Tabella* (*Attributo1*, ... , *AttributoN*)

Con entrambe le forme è possibile specificare che cosa fare in caso di cancellazione o di variazione del valore della chiave primaria associata alla chiave esterna, con la clausola:

ON { DELETE | UPDATE } { CASCADE | SET NULL | SET DEFAULT | RESTRICT | NO ACTION }

Il significato delle azioni da intraprendere è:

CASCADE l'aggiornamento di un valore della chiave primaria nella tabella primaria si riflette a catena sulle righe associate;

SET NULL permette l'eliminazione di un valore della chiave nella tabella primaria: i valori correlati nella tabella associata assumono il valore NULL;

SET DEFAULT permette l'eliminazione di un valore della chiave nella tabella primaria: i valori correlati nella tabella associata assumono un valore di default;

RESTRICT o NO ACTION
rifiuta le modifiche che violano un vincolo: impedisce di cancellare o modificare un valore della chiave nella tabella primaria per cui esistano righe correlate nella tabella associata; è il comportamento di default se non vengono espresse esplicitamente le clausole ON DELETE e ON UPDATE.

Loescher Editore - Vietata la vendita e la diffusione

Esempio 3.1

Creazione delle tabelle:
Fiumi con chiave primaria *NomeF*;
Nazioni con chiave primaria *NomeN* e chiavi esterne *Continente* e *Capitale*; il continente è obbligatorio; la capitale non può avere valori duplicati;
Attraversamento con chiave primaria {*Nazione*, *Fiume*} e chiavi esterne *Nazione* e *Fiume*.

```
CREATE TABLE Fiumi (
   NomeF CHARACTER(20) PRIMARY KEY,
   Lunghezza INTEGER
);

CREATE TABLE Nazioni (
   NomeN CHARACTER(20) PRIMARY KEY,
   Continente CHARACTER(2) NOT NULL,
   Capitale INTEGER UNIQUE,
   Superficie INTEGER,
   Popolazione INTEGER,
   FOREIGN KEY (Continente) REFERENCES Continenti(Sigla)
      ON UPDATE CASCADE,
   FOREIGN KEY (Capitale) REFERENCES Citta(IdCitta)
      ON DELETE SET NULL
      ON UPDATE CASCADE
);

CREATE TABLE Attraversamento (
   Nazione CHARACTER(20),
   Fiume CHARACTER(20),
   PRIMARY KEY (Nazione,Fiume),
   FOREIGN KEY (Nazione) REFERENCES Nazioni(NomeN)
      ON UPDATE CASCADE,
   FOREIGN KEY (Fiume) REFERENCES Fiumi(NomeF)
      ON UPDATE CASCADE
);
```

ASSERZIONI

Si possono specificare altri vincoli interrelazionali mediante le asserzioni.

CREATE ASSERTION *Asserzione* **CHECK** (*Condizione*);

La condizione deve sempre essere verificata; qualsiasi modifica alle relazioni che la renda falsa viene rifiutata dal DBMS.

A volte è necessario cambiare il tipo di controllo associato al vincolo, assegnandogli la modalità differita anziché quella immediata di default:

CREATE ASSERTION *Asserzione* CHECK (*Condizione*)
SET CONSTRAINTS *Vincolo* [IMMEDIATE | **DEFERRED**];

Il controllo differito è necessario in situazioni in cui non è possibile effettuare una modifica consistente al database con una sola operazione.

Loescher Editore - Vietata la vendita e la diffusione

Modifica di tabelle

Una tabella può essere modificata con il comando ALTER TABLE.

È possibile aggiungere o eliminare una colonna con il comando:

ALTER TABLE *Tabella*
{ **ADD COLUMN** *Attributo Dominio* [DEFAULT *ValorediDefault*] |
DROP COLUMN *Attributo* };

Nota

È anche possibile rinominare la tabella con la clausola RENAME TO e modificare il tipo di un campo con le clausole CHANGE e MODIFY.

Esempio 3.2

Inserimento nella tabella *Nazioni* della colonna *SiglaNazione*.

```
ALTER TABLE Nazioni
    ADD COLUMN SiglaNazione VARCHAR(3) UNIQUE;
```

È possibile aggiungere o eliminare un vincolo con il comando:

ALTER TABLE *Tabella*
{ **ADD CONSTRAINT** *Vincolo* |
DROP CONSTRAINT *Vincolo* };

Eliminazione di tabelle

Si può eliminare una tabella con il comando:

DROP TABLE *Tabella* [RESTRICT | CASCADE | SET NULL];

RESTRICT non permette la cancellazione se la tabella è legata da vincoli ad altre tabelle;
CASCADE cancella in cascata tutte le tabelle associate;
SET NULL pone a NULL tutti i valori delle chiavi esterne correlate in altre tabelle.

Esempio 3.3

Eliminazione della tabella *Mari*.

```
DROP TABLE Mari;
```

3.4 DEFINIZIONE DEGLI INDICI

Nota

Gli indici velocizzano l'accesso ai dati. Significato, organizzazione e modalità di scelta degli indici sono spiegati nel capitolo 6.

Si può creare un indice su una tabella usando il comando CREATE INDEX indicando il nome della tabella e il nome degli attributi su cui si vuole creare l'indice.
La clausola UNIQUE permette di specificare che l'indice non può contenere valori duplicati.

CREATE [**UNIQUE**] **INDEX** *Indice*
ON *Tabella* (*Attributo1*, ... , *AttributoN*);

Loescher Editore - Vietata la vendita e la diffusione

ESEMPIO 3.4

Creazione di indici:
sull'attributo *Mare* di *Litorale*;
sugli attributi {*NomeC*, *Nazione*} di *Città* con valori univoci.

```
CREATE INDEX Mare
   ON Litorale (Mare);

CREATE UNIQUE INDEX CittaNazione
   ON Citta (NomeC, Nazione);
```

Si può eliminare un indice con il comando:

DROP INDEX *Indice* **ON** *Tabella*;

3.5 MANIPOLAZIONE DEI DATI

Le operazioni manipolazione dei dati (DML) sono quelle relative all'inserimento dei dati nelle tabelle o alla loro modifica o cancellazione.

Per inserire una riga in una tabella si può usare il comando INSERT INTO specificando i valori degli attributi desiderati:

INSERT INTO *Tabella* [(*Attributo1*, ... , *AttributoN*)]
VALUES (*Valore1*, ... , *ValoreN*);

gli attributi non specificati assumono un valore di deafult specificato al momento della definizione della tabella o il valore NULL; se il valore NULL non è consentito a causa di un vincolo di obbligatorietà l'inserimento della riga viene rifiutato.
I valori vengono assegnati agli attributi in base all'ordine in cui compaiono nelle due liste; se la lista degli attributi non è specificata viene considerato l'ordine con cui sono definiti nella tabella.

ESEMPIO 3.5

Inserimento dei dati di alcuni laghi.

```
INSERT INTO Laghi (NomeL, Superficie) VALUES
   ('Huron', 59000),
   ('Lago di Garda', 370),
   ('Lago Superiore', 82000);
```

Si possono modificare i valori di una o più righe con il comando UPDATE:

UPDATE *Tabella*
SET *Attributo1* = { *Espressione1* | NULL | DEFAULT }
...
AttributoN = { *EspressioneN* | NULL | DEFAULT }
[**WHERE** *Condizione*];

Il valore di un attributo può essere il risultato di un'espressione, che può anche coinvolgere il valore corrente dell'attributo, il valore NULL o il valore di default specificato al momento della definizione della tabella.

È possibile modificare una riga individuandola mediante una condizione specificata nella clausola WHERE.

Loescher Editore - Vietata la vendita e la diffusione

ESEMPIO 3.6

Modifica della popolazione della città di Marsiglia.

```
UPDATE Citta
    SET Popolazione=807000
    WHERE NomeC='Marsiglia';
```

È possibile operare su gruppi di record se la clausola WHERE ha una condizione verificata su più righe.
Se la clausola WHERE non è presente la modifica viene eseguita su tutte le righe.

ESEMPIO 3.7

Correzione del nome del Danubio (scritto erroneamente in minuscolo) in tutte le righe della relazione *Attraversamento*.

```
UPDATE Attraversamento
    SET Fiume = 'Danubio'
    WHERE Fiume = 'danubio';
```

Si possono cancellare una o più righe della tabella con il comando DELETE:

DELETE FROM *Tabella*
[**WHERE** *Condizione*];

È possibile cancellare gruppi di record se la clausola WHERE ha una condizione verificata su più righe.
Se la clausola WHERE non è presente vengono cancellate tutte le righe della tabella.

ESEMPIO 3.8

Cancellazione dei dati del Mar Nero dalla tabella *Mari* e dei dati di tutte le città della Germania dalla tabella *Citta*.

```
DELETE FROM Mari
    WHERE NomeM = 'Mar Nero';

DELETE FROM Citta
    WHERE Nazione = 'Germania';
```

3.6 INTERROGAZIONI

SQL è un linguaggio non procedurale, cioè per effettuare interrogazioni non è necessario spiegare come reperire i dati ma solo quali sono i dati che interessano.

Per effettuare interrogazioni (**query**) si usa il comando SELECT.

FORMATO BASE DEL COMANDO SELECT

Il formato base del comando SELECT è:

SELECT *Attributo1*, ... , *AttributoN*
FROM *Tabella*
[**WHERE** *Condizione*];

Loescher Editore - Vietata la vendita e la diffusione

il comando agisce sulla tabella specificata nella clausola FROM e restituisce tutte le righe che soddisfano la condizione indicata nella clausola WHERE, fornendo solo i valori degli attributi elencati.

Se la clausola WHERE non è presente vengono restituite tutte le righe della tabella.
Se si desiderano tutti gli attributi è possibile usare il simbolo * al posto dell'elenco.

Esempio 3.9

Alcuni esempi di interrogazioni.

Tutti gli attributi dei laghi:

```
SELECT *
    FROM Laghi;
```

Nomi degli oceani:

```
SELECT NomeM
    FROM Mari
    WHERE Oceano = true;
```

Nome e popolazione delle città italiane:

```
SELECT NomeC, Popolazione
    FROM Citta
    WHERE Nazione = 'Italia';
```

La clausola WHERE corrisponde all'operazione di **selezione** delle righe che soddisfano la condizione specificata.

L'elenco di attributi dopo la parola SELECT corrisponde all'operazione di **proiezione** su quegli attributi.
Per realizzare effettivamente l'operazione di proiezione, che presenta una volta sola le righe duplicate, bisogna specificare il predicato DISTINCT dopo la parola SELECT.

Nota

Dopo la parola SELECT si possono specificare i predicati ALL o DISTINCT; ALL è il valore di default e restituisce tutte le righe che soddisfano la condizione specificata, DISTINCT indica che eventuali righe duplicate nel risultato devono essere presentate una volta sola.

Esempio 3.10

Nazioni in cui ci sono città con popolazione superiore a 2.000.000.

```
SELECT DISTINCT Nazione
    FROM Citta
    WHERE Popolazione > 2000000;
```

Oltre alle colonne presenti nella tabella è possibile inserire nel risultato valori prodotti dal calcolo di espressioni, cioè **attributi derivati**. È possibile specificare un'intestazione per la nuova colonna con la clausola **AS**.

Nota

Se la clausola AS non è presente il nome della colonna viene stabilito per default in modo generico dipendente dal DBMS utilizzato.

Esempio 3.11

Nome, superficie, popolazione e densità di ogni nazione.

Loescher Editore - Vietata la vendita e la diffusione

```
SELECT NomeN, Superficie, Popolazione, (Popolazione/Superficie) AS
Densita
    FROM Nazioni;
```

La clausola AS può essere utilizzata anche per ridenominare l'intestazione di una colonna esistente.

Esempio 3.12

Elenco dei nomi degli oceani dando all'elenco l'intestazione "Oceani".

```
SELECT NomeM AS Oceani
    FROM Mari
    WHERE Oceano = true;
```

Query parametriche

Le query possono essere **parametrizzate** in modo da poter utilizzare la stessa query per ricerche relative a valori diversi.

Si può parametrizzare una query facendola precedere dalla dichiarazione:

PARAMETERS *par1*[:*Tipo1*], ... , *parN*[:*TipoN*];

Al momento dell'interrogazione viene richiesto di immettere il valore di ciascun parametro e, se specificato, viene controllato il tipo.

Nota

Il modo in cui possono essere specificati i parametri in realtà dipende dal DBMS utilizzato; alcuni esempi sono proposti nei capitoli di MySQL e Access.

Creazione di tabelle temporanee

È possibile conservare il risultato di una query in una tabella temporanea aggiungendo all'istruzione SELECT la clausola INTO seguita dal nome da assegnare alla nuova tabella.

Esempio 3.13

Creazione di una tabella temporanea con nome e popolazione delle città italiane.

```
SELECT NomeC, Popolazione
    FROM Citta
    INTO PopolazioniCitta;
```

La condizione di ricerca

La condizione è un'espressione booleana che può utilizzare gli operatori relazionali e gli operatori logici.

Gli **operatori relazionali** sono:

=	uguale
!= o <>	diverso

>	maggiore
!> o <=	non maggiore cioè minore o uguale

<	minore
!< o >=	non minore cioè maggiore o uguale

Loescher Editore - Vietata la vendita e la diffusione

Gli **operatori logici** sono:

AND OR NOT

Tabella 3.3 *Tabella di verità degli operatori booleani*

X	Y	X and Y	X or Y	not X
falso	falso	falso	falso	vero
falso	vero	falso	vero	
vero	falso	falso	vero	falso
vero	vero	vero	vero	

Esempio 3.14

Nome e popolazione delle città italiane con popolazione inferiore a 1.000.000.

```
SELECT NomeC, Popolazione
    FROM Citta
    WHERE Nazione = 'Italia' AND Popolazione < 1000000;
```

La condizione relativa alla presenza di valori NULL in un campo si esprime con **IS NULL**.
La condizione IS NOT NULL verifica se nel campo è invece presente un valore.

Esempio 3.15

Nomi delle città per cui non è noto il valore di *Popolazione*.

```
SELECT NomeC
    FROM Citta
    WHERE Popolazione IS NULL;
```

Per ottenere i nomi delle città per cui il valore di *Popolazione* è noto basta usare l'operatore NOT:

```
SELECT NomeC
    FROM Citta
    WHERE Popolazione IS NOT NULL;
```

Nei confronti inoltre si possono usare gli operatori BETWEEN, IN e LIKE.

BETWEEN permette di controllare se un valore è compreso in un certo intervallo di valori (estremi inclusi); l'operatore NOT permette di specificare la condizione che il valore sia esterno all'intervallo.

Esempio 3.16

Nazioni con superficie compresa tra 300.000 e 600.000.

```
SELECT NomeN
    FROM Nazioni
    WHERE Superficie BETWEEN 300000 AND 600000;
```

Si può ottenere lo stesso risultato anche usando una condizione che utilizza solo operatori relazionali e logici:

```
SELECT NomeN
    FROM Nazioni
    WHERE Superficie >= 300000 AND Superficie <= 600000;
```

IN permette di controllare se un valore appartiene a un elenco di valori separati da virgole, specificato tra parentesi tonde; l'operatore NOT permette di specificare la non appartenenza all'elenco di valori.

Loescher Editore - Vietata la vendita e la diffusione

Esempio 3.17

Città appartenenti all'Italia, alla Francia o alla Germania.

```
SELECT NomeC
    FROM Citta
    WHERE Nazione IN ('Italia','Francia','Germania');
```

LIKE permette di effettuare confronti di stringhe; nella stringa di confronto possono comparire i caratteri speciali _ che corrisponde a un solo carattere arbitrario e % che corrisponde a un gruppo di qualsiasi numero di caratteri arbitrari.

Esempio 3.18

Nomi delle nazioni che cominciano con la lettera "A".

```
SELECT NomeN
    FROM Nazioni
    WHERE NomeN LIKE 'A%';
```

Ordinamenti

Si possono ordinare i risultati di una interrogazione usando la clausola **ORDER BY**.
Nella clausola ORDER BY si possono specificare uno o più attributi indicandone il nome o, per quelli elencati dopo la parola SELECT, la posizione a partire da 1 .
L'ordinamento può essere crescente (valore di default o **ASC**) o decrescente specificando **DESC**.
Il valore NULL viene considerato all'inizio delle sequenze crescenti.
La clausola ORDER BY, se presente, deve essere l'ultima di un comando SELECT.

Esempio 3.19

Alcuni esempi di ordinamento.

Elenco delle nazioni in ordine crescente di superficie:

```
SELECT NomeN, Superficie
    FROM Nazioni
    ORDER BY Superficie ASC;
```

Elenco delle città in ordine di *Nazione* e in ordine alfabetico all'interno della stessa nazione:

```
SELECT NomeC, Nazione
    FROM Citta
    ORDER BY Nazione, NomeC;
```

Elenco delle nazioni in ordine decrescente di densità:

```
SELECT NomeN, Popolazione/Superficie
    FROM Nazioni
    ORDER BY 2 DESC;
```

Funzioni di aggregazione

Nel comando SELECT si possono specificare funzioni predefinite, chiamate **funzioni di aggregazione**, che agiscono su una colonna di un insieme di righe e restituiscono un valore; le righe considerate sono quelle ottenute dalla selezione specificata nella clausola WHERE.

Loescher Editore - Vietata la vendita e la diffusione

COUNT

La funzione **COUNT**() conta il numero di righe selezionate dalla query; come parametro è possibile specificare:

(*)	per contare tutte le righe della tabella;
(*Attributo*)	per contare le righe della tabella in cui il valore dell'attributo non è NULL.

ESEMPIO 3.20

Esempi di conteggi.

Numero delle città:

```
SELECT COUNT(*)
    FROM Citta;
```

Numero delle città italiane:

```
SELECT COUNT(*)
    FROM Citta
    WHERE Nazione = 'Italia';
```

Numero delle città italiane per cui è noto il valore di *Popolazione*:

```
SELECT COUNT(Popolazione)
    FROM Citta
    WHERE Nazione = 'Italia';
```

L'ultima query è equivalente a:

```
SELECT COUNT(NomeC)
    FROM Citta
    WHERE Nazione = 'Italia' AND Popolazione IS NOT NULL;
```

La parola **DISTINCT** specificata prima del nome dell'attributo individua il numero di valori diversi tra loro presenti nella colonna *Attributo.*

Nota

La clausola DISTINCT non può essere usata nel formato con l'asterisco (COUNT(*)); non avrebbe senso perché le righe della tabella sono già tutte diverse tra loro per definizione.

ESEMPIO 3.21

Numero delle nazioni per cui è stata inserita almeno una città.

```
SELECT COUNT(DISTINCT Nazione)
    FROM Citta;
```

SUM

La funzione **SUM**() somma i valori non NULL di un attributo numerico specificato come argomento della funzione; si può usare DISTINCT prima del nome dell'attributo per considerare solo una volta i valori ripetuti. L'argomento della funzione può essere anche un'espressione numerica che ha come operandi attributi di tipo numerico.

ESEMPIO 3.22

Calcolo della popolazione europea.

Loescher Editore - Vietata la vendita e la diffusione

```
SELECT SUM(Popolazione)
    FROM Nazioni
    WHERE Continente = 'EU';
```

AVG

La funzione **AVG**() calcola la media dei valori non NULL di un attributo numerico della tabella; si può usare DISTINCT prima del nome dell'attributo per considerare solo una volta i valori ripetuti.
L'argomento della funzione può essere anche un'espressione numerica che ha come operandi attributi di tipo numerico.

Esempio 3.23

Densità media di popolazione delle nazioni europee.

```
SELECT AVG(Popolazione/Superficie) AS DensitaMedia
    FROM Nazioni
    WHERE Continente = 'EU';
```

MIN e MAX

Le funzioni **MIN**() e **MAX**() calcolano rispettivamente il minimo e il massimo dei valori non NULL di un attributo (numerico o stringa).
L'argomento della funzione può essere anche un'espressione numerica.

Esempio 3.24

Superficie massima e densità minima di popolazione delle nazioni.

```
SELECT MAX(Superficie), MIN(Popolazione/Superficie)
    FROM Nazioni;
```

Raggruppamenti

È possibile applicare le funzioni di aggregazione a gruppi di righe individuate dalla clausola **GROUP BY**.
In pratica GROUP BY permette di raggruppare le righe in base al valore di uno o più attributi.
Le funzioni di aggregazione vengono applicate alle righe di ciascun gruppo, producendo un risultato per ciascun gruppo.
I valori NULL vengono raggruppati, ma sono considerati solo dalla funzione COUNT.
Quando si utilizza la clausola GROUP BY, tutti gli attributi specificati accanto a SELECT che non sono argomenti di una funzione di aggregazione devono essere inclusi nella clausola GROUP BY.

Esempio 3.25

Popolazione delle città più popolosa e meno popolosa di ogni nazione.

```
SELECT Nazione, MAX(Popolazione), MIN(Popolazione)
    FROM Citta
    GROUP BY Nazione;
```

La clausola **HAVING** permette di specificare una condizione sui gruppi creati con GROUP BY.

Loescher Editore - Vietata la vendita e la diffusione

Esempio 3.26

Media della popolazione delle città per le nazioni in cui la media è superiore a 1.500.000.

```
SELECT Nazione, AVG(Popolazione) AS MediaPop
    FROM Citta
    GROUP BY Nazione
    HAVING MediaPop > 1500000;
```

Clausole del comando SELECT

SELECT	elenco degli attributi
FROM	tabella da cui estrarre le righe
WHERE	condizioni di selezione
GROUP BY	attributi da considerare per i raggruppamenti
HAVING	condizioni sui raggruppamenti
ORDER BY	ordinamenti sugli attributi

Le clausole, se presenti, devono essere espresse in questo ordine; le uniche obbligatorie sono SELECT e FROM.

L'ordine con cui vengono elaborate le clausole è FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY.

Operazione di join

Il comando SELECT può essere utilizzato per effettuare operazioni di **join** su più tabelle. La forma per un join su due tabelle è:

SELECT *Attributi*
    FROM ***Tabella1***, ***Tabella2***
    WHERE ***CondizioneDiJoin***;

Nota

Se la clausola WHERE non è specificata il comando corrisponde al prodotto cartesiano delle tabelle. Un join può essere visto come un prodotto cartesiano seguito da operazioni di selezione. In realtà il DBMS, se possibile, anticipa le selezioni per ottimizzare l'esecuzione dell'interrogazione.

La condizione di join più comune è quella di uguaglianza relativa a un **equi-join**.

Esempio 3.27

Nome delle nazioni e continente di appartenenza.

```
SELECT NomeN, NomeCon
    FROM Nazioni, Continenti
    WHERE Continente = Sigla;
```

Nel **join naturale** gli attributi da correlare hanno lo stesso nome.

Se nelle tabelle utilizzate ci sono attributi con lo stesso nome è possibile specificare l'attributo a cui ci si riferisce con la notazione puntata *NomeTabella.NomeAttributo*.

Loescher Editore - Vietata la vendita e la diffusione

Se la sigla del continente si chiamasse *SiglaCon* sia nella tabella *Nazioni* che nella tabella *Continenti* si scriverebbe:

```
SELECT NomeN, NomeCon
    FROM Nazioni, Continenti
    WHERE Nazioni.SiglaCon = Continenti.SiglaCon;
```

Per rendere più comprensibile la query è possibile specificare in ogni caso i nomi delle tabelle con la notazione puntata, anche se i nomi degli attributi sono diversi tra loro.

La query dell'esempio 3.27 può essere resa più comprensibile scrivendo:

```
SELECT NomeN, NomeCon
    FROM Nazioni, Continenti
    WHERE Nazioni.Continente = Continenti.Sigla;
```

Per rendere più semplice la scrittura è possibile assegnare degli alias ai nomi delle tabelle con la clausola AS.

```
SELECT NomeN, NomeCon
    FROM Nazioni AS N, Continenti AS C
    WHERE N.Continente = C.Sigla;
```

Per realizzare i **self-join** è necessario ricorrere all'uso degli alias: la relazione deve essere ridenominata con alias che ricordino i due diversi ruoli.

Esempio 3.28

Elenco dei fiumi che sono affluenti di altri, con la lunghezza dei due fiumi.

```
SELECT Affluente, F1.Lunghezza, Fiume, F2.Lunghezza
    FROM Affluenza, Fiumi AS F1, Fiumi AS F2
    WHERE Affluente = F1.NomeF AND Fiume = F2.NomeF;
```

Traduzione in SQL di espressioni in algebra relazionale

Le espressioni dell'algebra relazionale possono sempre essere realizzate mediante comandi SQL.

Esempio 3.29

Comando SQL che realizza l'espressione dell'algebra relazionale dell'esempio 2.7 (nome, popolazione, capitale e popolazione della capitale di tutte le nazioni dell'Europa).

$$\Pi_{\text{NomeN, Nazioni.Popolazione, NomeC, Città.Popolazione}} ((\sigma_{\text{NomeCon='Europa'}} (\text{Nazioni} \bowtie_{\text{Continente=Sigla}} \text{Continenti})) \bowtie_{\text{Capitale=IdCittà}} \text{Città})$$

```
SELECT NomeN, Nazioni.Popolazione, Citta.NomeC, Citta.Popolazione
    FROM Nazioni, Continenti, Citta
    WHERE NomeCon='Europa' AND Continente=Sigla AND Capitale=IdCitta;
```

Esempio 3.30

Comando SQL che realizza l'espressione dell'algebra relazionale dell'esempio 2.8 (nome e superficie delle nazioni che sono attraversate dal Danubio).

Loescher Editore - Vietata la vendita e la diffusione

$\Pi_{NomeN,\ Superficie}\ ((\sigma_{Fiume='Danubio'}\ (Attraversamento)) \bowtie_{Attraversamento.Nazione=Nazioni.NomeN} Nazioni)$

```
SELECT NomeN, Superficie
    FROM Attraversamento, Nazioni
    WHERE Fiume = 'Danubio' AND NomeN = Nazione;
```

Sintassi del join in SQL-2

L'equi-join in SQL-2 viene indicato come **inner join**.

SELECT *Attributi*
FROM *Tabella1*
INNER JOIN *Tabella2*
ON *Tabella1.Attributo* = *Tabella2.Attributo*;

La parola INNER è il valore di default per il join e quindi può essere omessa.

Il **join naturale** può essere codificato in alternativa con:

SELECT *Attributi*
FROM *Tabella1*
INNER JOIN *Tabella2*
USING *Attributo*;

SELECT *Attributi*
FROM *Tabella1*
NATURAL JOIN *Tabella2*;

Esempio 3.31

Nome delle nazioni e continente di appartenenza.

```
SELECT NomeN, NomeCon
    FROM Nazioni
    INNER JOIN Continenti
    ON Continente = Sigla;
```

Se la sigla del continente si chiamasse *SiglaCon* sia nella tabella *Nazioni* che nella tabella *Continenti* si potrebbe scrivere:

```
SELECT NomeN, NomeCon
    FROM Nazioni
    NATURAL JOIN Continenti;
```

La forma SQL-2 dell'istruzione è più chiara perché separa la condizione di giunzione (clausola ON) dalle condizioni di selezione (clausola WHERE).

Per maggior chiarezza nella sintassi SQL-1 è opportuno scrivere la clausola WHERE su più righe riportando prima le condizioni per il join e poi quelle per le selezioni.

Esempio 3.32

Nomi delle nazioni con superficie inferiore a 1000 con il continente di appartenenza.

Sintassi SQL-2:

```
SELECT NomeN, NomeCon
    FROM Nazioni
    JOIN Continenti
    ON Continente = Sigla
    WHERE Superficie < 1000;
```

Loescher Editore - Vietata la vendita e la diffusione

Sintassi SQL-1:

```
SELECT NomeN, NomeCon
    FROM Nazioni, Continenti
    WHERE Continente = Sigla
        AND Superficie < 1000;
```

SQL-2 offre anche le clausole per realizzare **join esterni**: LEFT JOIN, RIGHT JOIN e FULL JOIN.

SELECT *Attributi*
FROM *Tabella1*
{ **LEFT** | **RIGHT** | **FULL** } **JOIN** *Tabella2*
ON *Tabella1.Attributo* = *Tabella2.Attributo*;

Esempio 3.33

Esempi di join esterni.

Nota Le situazioni seguenti sono possibili solo se non sono stabiliti vincoli di integrità referenziale sulle relazioni *Citta* e *Nazioni*.

```
SELECT NomeC, Nazione
    FROM Citta
    LEFT JOIN Nazioni
    ON Nazione = NomeN;
```

Il join sinistro tra *Citta* e *Nazioni* sul nome della nazione elenca tutte le città (anche quelle per cui non esiste la nazione corrispondente nella tabella *Nazioni*), ma non elenca le nazioni per cui non esiste la capitale corrispondente nella tabella *Citta*.

```
SELECT NomeN, Capitale
    FROM Citta
    RIGHT JOIN Nazioni
    ON IdCitta = Capitale;
```

Il join destro tra *Citta* e *Nazioni* sull'identificatore della capitale elenca tutte le nazioni (anche quelle per cui non esiste la capitale nella tabella *Citta*), ma non elenca le città per cui non esiste la nazione corrispondente nella tabella *Nazioni*.

```
SELECT NomeC, Nazione
    FROM Citta
    FULL JOIN Nazioni
    ON Nazione = NomeN;
```

Il full join tra *Citta* e *Nazioni* elenca tutte le città e tutte le nazioni (anche quelle per cui non esiste una corrispondenza nell'altra tabella).

Si possono utilizzare i join esterni per verificare se ci sono in due tabelle dati non corrispondenti.

Ciò si può verificare se non sono stati definiti vincoli di integrità referenziale.

Esempio 3.34

Controllare se ci sono città appartenenti a nazioni di cui non sono ancora stati inseriti i dati nella tabella *Nazioni*.

Loescher Editore - Vietata la vendita e la diffusione

```
SELECT NomeC
    FROM Citta
    LEFT JOIN Nazioni
    ON Nazione = NomeN
    WHERE NomeN IS NULL;
```

Nota

Per realizzare un theta join, in ciascun comando invece della condizione per uguale può essere indicata anche una condizione per >, < o <>.

Si possono realizzare anche join tra più di due tabelle.

Esempio 3.35

Elenco delle nazioni con nome, continente, superficie, popolazione, capitale e popolazione della capitale.

```
SELECT NomeN, NomeCon, Superficie, N.Popolazione, NomeC AS
Capitale, C.Popolazione AS PopolazioneCapitale
    FROM Nazioni AS N
    JOIN Continenti
    ON Continente = Sigla
    JOIN Citta AS C
    ON Capitale = IdCitta;
```

Interrogazioni nidificate o sottoquery

Nella condizione della clausola WHERE, al posto di una costante, è possibile usare una clausola SELECT (**interrogazione nidificata** o **sottoquery**).

Nota

La sottoquery deve apparire come secondo termine nei confronti, mai come primo.

È possibile avere anche più livelli di nidificazione.

Query nidificate di tipo scalare

Una sottoquery che restituisce un **valore scalare** deve utilizzare funzioni di aggregazione o restituire un solo attributo di una sola riga.

SELECT *Attributi*
FROM *Tabella1*
WHERE *Attributo* { = | > | < | >= | <= | <> }
    (SELECT { Funzione | Attributo }
    FROM *Tabella2*
    WHERE *Condizione* **)**;

La query interna viene sempre eseguita prima della query esterna.

Esempio 3.36

Individuazione della nazione con superficie maggiore.

```
SELECT NomeN
FROM Nazioni
```

Loescher Editore - Vietata la vendita e la diffusione

```
WHERE Superficie =
    ( SELECT MAX(Superficie)
      FROM Nazioni
    );
```

Questo problema deve essere risolto usando un'interrogazione nidificata perché le funzioni di aggregazione non possono comparire nella clausola WHERE (possono comparire solo nelle clausole SELECT e HAVING).

Esempio 3.37

Elenco delle nazioni per cui la popolazione della capitale è inferiore alla popolazione media delle città, con popolazione della capitale e scarto rispetto alla popolazione media delle città.

```
SELECT Nazione, C.Popolazione,
    (( SELECT AVG(Popolazione) FROM CITTA ) - C.Popolazione) AS Scarto
    FROM Nazioni
    INNER JOIN Citta AS C
    ON Capitale = IdCitta
    WHERE C.Popolazione <
        ( SELECT AVG(Popolazione)
          FROM Citta
        );
```

Esempio 3.38

Elenco delle nazioni che sono bagnate da almeno due mari, con i nomi dei mari che le bagnano.

```
SELECT L.Nazione, L.Mare
FROM Litorale AS L
WHERE L.Nazione IN
    ( SELECT Nazione
      FROM Litorale
      GROUP BY Nazione
      HAVING COUNT(*) > 1
    )
ORDER BY L.Nazione;
```

Una query di questo tipo può essere utile in generale per sapere se un valore è presente più di una volta in una colonna e in tal caso in corrispondenza di quali righe.

Query nidificate di tipo non scalare

Si possono usare **sottoquery che producono un elenco di valori** inserendo nella condizione, dopo l'operatore relazionale di confronto, uno dei predicati **ANY** o **ALL**.

SELECT *Attributi*
FROM *Tabella1*
WHERE *Attributo* { = | > | < | >= | <= | <> } { **ANY** | **ALL** }
    (SELECT *Attributo*
    FROM *Tabella2*
    WHERE *Condizione* **)**;

Loescher Editore - Vietata la vendita e la diffusione

Normalmente il confronto viene eseguito per un solo attributo: la query interna restituisce il valore di un attributo per più righe.

È possibile anche considerare più attributi, verificando una condizione su più attributi elencati tra parentesi e lo stesso numero di attributi restituiti dalla sottoquery per più righe. **Nota**

Anche in questo caso si può pensare che la query interna venga sempre eseguita prima della query esterna.

In realtà il modo in cui le operazioni vengono eseguite può essere diverso, per motivi di efficienza. **Nota**

Sottoquery e tabelle temporanee

Invece di usare query nidificate si possono usare due query in successione: si esegue la query interna e si salva il risultato in una tabella temporanea, poi si esegue la query esterna utilizzando il risultato temporaneo.

I predicati ANY e ALL permettono di effettuare un confronto tra un attributo e un insieme di valori:

- **ANY** dà come risultato vero se il confronto è vero per almeno uno dei valori dell'elenco, falso se la sottoquery restituisce un insieme vuoto o se il confronto è falso per tutti i valori dell'elenco;

Esempio 3.39

Nome delle città della Francia con popolazione superiore a quella di almeno una città della Germania.

```
SELECT NomeC
FROM Citta
WHERE Nazione = 'Francia' AND Popolazione > ANY
  ( SELECT Popolazione
    FROM Citta
    WHERE Nazione = 'Germania'
);
```

- **ALL** dà come risultato vero se il confronto è vero per tutti i valori dell'elenco, falso se il confronto è falso per almeno uno dei valori dell'elenco.

Esempio 3.40

Nome delle città della Francia con popolazione superiore a quella di tutte le città della Germania.

```
SELECT NomeC
FROM Citta
WHERE Nazione = 'Francia' AND Popolazione > ALL
  ( SELECT Popolazione
    FROM Citta
    WHERE Nazione = 'Germania'
);
```

Loescher Editore - Vietata la vendita e la diffusione

Equivalenza tra ANY o ALL e IN

attributo = ANY(SELECT...)	equivale a	Attributo IN (SELECT ...)
attributo <> ALL(SELECT...)	equivale a	Attributo NOT IN (SELECT ...)

Query correlate

Si dicono query correlate le query in cui la sottoquery agisce su una tabella presente nella query esterna. Se la sottoquery indica nella clausola FROM la tabella presente nella query esterna, risulta necessario utilizzare un alias.

SELECT *Attributi*
FROM *Tabella1*
WHERE *Attributo* { = | > | < | <= | <= | <> } [ANY | ALL]
    (SELECT *Attributo*
    FROM *Tabella2*
    WHERE *Tabella2.Attributo* { = | > | < | <= | <= | <> } *Tabella1.Attributo* **)**;

Esempio 3.41

Nome delle nazioni che hanno superficie superiore alla superficie media delle nazioni del continente a cui appartengono.

```
SELECT NomeN
FROM Nazioni AS N1
WHERE Superficie >
    ( SELECT AVG(Superficie)
      FROM Nazioni AS N2
      WHERE N2.Continente = N1.Continente);
```

Una query correlata può essere interpretata in questo modo: per ogni riga della query esterna viene valutata la query interna e viene eseguito il confronto.

Nota Per le query correlate non vale l'interpretazione che la query interna sia eseguita sempre prima della query esterna.

L'operatore EXISTS

L'operatore logico **EXISTS** dà come risultato vero se la sottoquery restituisce almeno un valore, falso se non restituisce nessun valore (insieme vuoto).

SELECT *Attributi*
FROM *Tabella1*
WHERE [NOT] **EXISTS**
    (SELECT *
    FROM *Tabella2*
    WHERE *Tabella2.Attributo* { = | > | < | <= | <= | <> } *Tabella1.Attributo* **)**;

Nota L'operatore EXISTS è utilizzabile in modo significativo solo nelle query correlate.

Nota Gli attributi specificati nella sottoquery usata col predicato EXISTS sono irrilevanti, quindi di solito si usa l'asterisco.

Loescher Editore - Vietata la vendita e la diffusione

Esempio 3.42

Nome e continente delle nazioni che non sono bagnate da nessun mare.

```
SELECT NomeN, NomeCon
    FROM Nazioni JOIN Continenti
    ON Continente = Sigla
    WHERE NOT EXISTS
    ( SELECT *
      FROM Litorale
      WHERE Nazione = NomeN
    );
```

Esempio 3.43

Elenco delle città con popolazione maggiore o uguale a 1.000.000 per ogni nazione in cui esiste almeno una città con popolazione inferiore a 10.000.

```
SELECT C1.NomeC, C1.Popolazione
FROM Citta AS C1
WHERE C1.Popolazione >= 1000000
AND EXISTS
    ( SELECT *
      FROM Citta AS C2
      WHERE C1.Nazione = C2.Nazione
      AND C2.Popolazione < 10000
    );
```

La query può essere riscritta come self-join:

```
SELECT DISTINCT C1.NomeC, C1.Popolazione
    FROM Citta AS C1 JOIN Citta AS C2
    ON C1.Nazione = C2.Nazione
    WHERE C1.Popolazione >= 1000000 AND C2.Popolazione < 10000;
```

Unione, intersezione e differenza

Unione, intersezione e differenza possono essere realizzate rispettivamente con i comandi **UNION**, **INTERSECT** e **EXCEPT**.

Tabella1 **UNION** *Tabella2;*

Tabella1 **INTERSECT** *Tabella2;*

Tabella1 **EXCEPT** *Tabella2;*

Nota

Le tabelle devono essere compatibili.

Gli operandi, oltre che tabelle, possono essere risultati di istruzioni SELECT:

***SelectSQL*1**
UNION
SelectSQL2;

Loescher Editore - Vietata la vendita e la diffusione

Se il FULL JOIN non è disponibile è possibile realizzarlo come unione di LEFT JOIN e RIGHT JOIN.

Le operazioni di **intersezione** e **differenza** possono essere realizzate anche usando **interrogazioni nidificate**.

Tabella1 INTERSECT *Tabella2*; equivale a:

```
SELECT *
FROM Tabella1
WHERE Chiave IN
    (SELECT Chiave
    FROM Tabella2);
```

e

Tabella1 EXCEPT *Tabella2*; equivale a:

```
SELECT *
FROM Tabella1
WHERE Chiave NOT
IN
    (SELECT Chiave
    FROM Tabella2);
```

Esempio 3.44

Nazioni dell'Europa che sono bagnate dal mare (esempio 2.9).

$(\Pi_{NomeN} (\sigma_{NomeCon='Europa'} (Nazioni \bowtie_{Continente=Sigla} Continenti))) \cap (\Pi_{Nazione} (Litorale))$

```
SELECT NomeN
    FROM Nazioni
    JOIN Continenti
    ON Continente=Sigla
    WHERE NomeCon='Europa' AND NomeN IN
        ( SELECT Nazione
          FROM Litorale
        );
```

Esempio 3.45

Nazioni dell'Europa che non sono bagnate dal mare (esempio 2.10).

$(\Pi_{NomeN} (\sigma_{NomeCon='Europa'} (Nazioni \bowtie_{Continente=Sigla} Continenti))) - (\Pi_{Nazione} (Litorale))$

```
SELECT NomeN
    FROM Nazioni
    JOIN Continenti
    ON Continente=Sigla
    WHERE NomeCon='Europa' AND NomeN NOT IN
        ( SELECT Nazione
          FROM Litorale
        );
```

Loescher Editore - Vietata la vendita e la diffusione

Uso di interrogazioni in comandi di manipolazione

Si possono combinare comandi di manipolazione con comandi di interrogazione che agiscono anche su altre tabelle.

INSERT INTO *Tabella* [(*Attributo1*, ... , *AttributoN*)]
SelectSQL;

UPDATE *Tabella*
SET *Attributo1* = ***SelectSQL***
...
AttributoN = ...
[**WHERE** *Condizione con Select nidificate*];

DELETE FROM *Tabella*
[**WHERE** *Condizione con Select nidificate*];

Esempio 3.46

Inserimento nella tabella *Citta* delle città contenute nella tabella *NuoveCitta* (che contiene per esempio le città di una nazione non ancora gestita).

```
INSERT INTO Citta (NomeC, Nazione)
    SELECT NC.NuovaC, NC.Nazione
    FROM NuoveCitta AS NC;
```

Esempio 3.47

Cancellazione dalla tabella *Nazioni* di tutte le nazioni dell'America (per esempio perché sono state separate in una diversa tabella).

```
DELETE FROM Nazioni
WHERE Continente IN
    ( SELECT Sigla
      FROM Continenti
      WHERE NomeCon LIKE '%America%'
    );
```

3.7 VISTE

Si possono creare delle viste logiche sui dati contenuti in una o più tabelle con il comando CREATE VIEW contenente un comando SELECT:

CREATE VIEW *NomeVista* **AS**
SelectSQL;

Esempio 3.48

Creazione di una vista con le sole città italiane.

Loescher Editore - Vietata la vendita e la diffusione

```
CREATE VIEW CittaItaliane AS
    SELECT NomeC, Popolazione
    FROM Citta
    WHERE Nazione='Italia';

SELECT * FROM CittaItaliane;
```

Si può eliminare una vista col comando:

DROP VIEW *NomeVista*;

Tramite una vista è possibile concedere una visione solo parziale dei dati, concentrando l'attenzione su questi ed eliminando il rischio di modifiche indesiderate sugli altri dati.

La vista appare all'utente come una normale tabella su cui è possibile operare con gli usuali comandi.

Se i dati reali vengono aggiornati, il contenuto della vista cambia poiché è determinato dinamicamente. È vero anche il viceversa: ogni modifica sulla vista si riflette anche sulle tabelle reali.
Non tutte le viste però sono aggiornabili: le viste che contengono risultati di funzioni di aggregazione non possono essere utilizzate per effettuare modifiche.

Differenza tra le viste logiche e le tabelle temporanee

Le tabelle temporanee una volta create sono indipendenti dai dati di partenza; modifiche sui dati reali e sulla tabella temporanea sono indipendenti tra loro.

Le viste logiche sono calcolate dinamicamente: modifiche sui dati reali e sulle viste sono intercambiabili.

Le viste logiche possono semplificare la scrittura di query complesse.

Esempio 3.49

Trovare la nazione con la superficie totale dei laghi maggiore.
Calcolare il rapporto medio tra superficie dei laghi e superficie della nazione.

Per semplificare le interrogazioni è possibile creare una vista che elenca per ogni nazione il numero di laghi e la superficie totale dei laghi:

```
CREATE VIEW LaghiNazione AS
   SELECT Nazione, COUNT(*) AS NrLaghi,
      SUM(Superficie) AS SuperficieTotale
   FROM Laghi
   JOIN Presenza
   ON NomeL = Lago
   GROUP BY Nazione;
```

Nazione con la superficie totale dei laghi maggiore:

```
SELECT Nazione
FROM LaghiNazione
WHERE SuperficieTotale =
    ( SELECT MAX(SuperficieTotale)
      FROM LaghiNazione
    );
```

Loescher Editore - Vietata la vendita e la diffusione

Rapporto medio tra superficie dei laghi e superficie della nazione:

```
SELECT AVG(LN.SuperficieTotale/N.Superficie)
    FROM Nazioni AS N
    JOIN LaghiNazione AS LN
    ON NomeN = Nazione;
```

3.8 TRIGGER

I **trigger**, detti anche **regole attive**, specificano il comportamento da assumere in situazioni prestabilite.
I trigger sono associati a una tabella e seguono il paradigma **Evento-Condizione-Azione** (ECA): ogni trigger si attiva quando si verifica uno specifico evento sulla tabella; se è soddisfatta la condizione specificata, viene eseguita l'azione stabilita.

Nota: Un trigger può causare eventi che azionano altri trigger (catene di attivazione), a volte anche in modo ciclico.

I trigger sono utili in molte situazioni ma sono complessi e sono implementati in modo differente dai DBMS.
Anche la sintassi standard dei trigger è in evoluzione; la sintassi in SQL-3 è:

CREATE TRIGGER *NomeTrigger*
[**BEFORE** | **AFTER**] [**INSERT** | **UPDATE** | **DELETE**] [OF *Attributi*] **ON** *Tabella*
[REFERENCING
    OLD ROW AS *NomeVecchiaRiga*
    NEW ROW AS *NomeNuovaRiga*]
[FOR EACH ROW]
WHEN *Condizione*
istruzioniSQL;

Il comando per la creazione di un trigger è un composto di tre parti:

- definizione dell'**evento** che attiva il controllo, di solito un'operazione di aggiornamento del database (inserimento, modifica o cancellazione); ogni trigger è sensibile a un solo evento; le parole BEFORE e AFTER permettono di specificare se il trigger deve attivarsi prima o dopo l'esecuzione dell'operazione;
- definizione, mediante la clausola WHEN, della **condizione** che deve essere soddisfatta;
- definizione dell'**azione** da eseguire.

La dichiarazione FOR EACH ROW precisa se l'azione deve essere eseguita per ogni riga modificata (altrimenti viene eseguita una sola volta indipendentemente dal numero di righe modificate); NEW ROW AS e OLD ROW AS permettono di indicare dei nomi con cui fare riferimento nella condizione e nell'azione ai dati prima e dopo l'aggiornamento.

Nota: Ovviamente NEW ROW non ha senso in operazioni di cancellazione e OLD ROW in operazioni di inserimento.

Nota: In alcune implementazioni per fare riferimento ai dati prima e dopo l'aggiornamento possono essere disponibili le variabili predefinite NEW e OLD.

Loescher Editore - Vietata la vendita e la diffusione

Alcuni **usi comuni** dei trigger sono:

- gestione di **vincoli di integrità** generici: poiché i trigger sono sensibili a tutti gli eventi che possono introdurre violazioni nei vincoli, permettono di verificare se c'è una violazione ed eventualmente intervenire; possono anche riparare il problema, non solo rifiutare l'operazione come avviene per i vincoli definiti mediante la clausola CHECK o le asserzioni;
- creazione di **file di log** delle operazioni effettuate;
- attuazione di qualsiasi meccanismo che deve essere guidato dal verificarsi di un evento.

Nota

La sintassi dei trigger può variare in base al DBMS. Esempi di trigger sono proposti nel capitolo di MySQL.

3.9 PROCEDURE

SQL-2 permette di definire procedure che vengono memorizzate all'interno del database come parte dello schema (**stored procedure**).
È possibile associare un nome a un'istruzione SQL, creando in pratica un nuovo comando.
La definizione della procedura può specificare dei parametri formali a cui deve essere associato un valore al momento del richiamo della procedura.

I principali **vantaggi** dell'uso di procedure sono:

- l'aumento della **sicurezza**, perché si possono costringere gli utenti ad accedere ai dati solo tramite procedure predefinite, limitando le operazioni che si possono fare e garantendo un maggior controllo;
- la maggior **semplicità nello sviluppo di applicazioni** che interagiscono col database che risultano più comprensibili e di più facile manutenzione;
- un **incremento delle prestazioni** perché occorre trasmettere meno informazioni tra client e server, riducendo il traffico sulla rete (le procedure però aumentano il carico di lavoro del server).

Lo standard SQL-2 prevede solo la definizione di procedure composte da un singolo comando SQL ma le implementazioni dei DBMS in genere forniscono estensioni che permettono di dichiarare variabili locali e di usare strutture di controllo.

Nota

Esempi di stored procedure sono proposti nel capitolo di MySQL.

3.10 CONTROLLO DELL'ACCESSO

L'amministratore del database può stabilire **i diritti di accesso** che gli utenti hanno sulle tabelle.

Il comando **GRANT** consente di specificare il tipo di accesso (*Privilegi*), le tabelle su cui l'accesso è consentito e l'elenco degli utenti a cui l'accesso è concesso (il valore *PUBLIC* indica che il privilegio specificato è concesso a tutti).

GRANT *Privilegi* **ON** *Tabella* **TO** [*Utenti* | PUBLIC];

I privilegi principali sono:

SELECT	accesso in lettura;
INSERT	inserimento di dati;

Loescher Editore - Vietata la vendita e la diffusione

UPDATE	modifica dei dati;
DELETE	cancellazione di righe;
ALL	tutti i permessi.

I permessi possono essere revocati con il comando **REVOKE**:

REVOKE *Privilegi* ON *Tabella* FROM *Utenti* [*Utenti* | PUBLIC];

I permessi possono essere assegnati, oltre che alle tabelle, anche alle **viste**.
Le viste logiche possono quindi essere utilizzate come strumento di protezione dei dati.

Ruoli in SQL-3

In SQL-3 il modello di controllo dell'accesso è di tipo **RBAC** (Role Based Access Control): i permessi di accesso sono determinati dai **ruoli** che gli utenti assumono; i ruoli corrispondono alle funzioni aziendali; gli utenti possono assumere più ruoli.

Si può creare un ruolo tramite il comando:

CREATE ROLE *NomeRuolo*;

Al ruolo si attribuiscono i privilegi tramite il comando GRANT.

Il comando GRANT viene utilizzato anche per concedere agli utenti la possibilità di ricoprire un certo ruolo.
Per fruire dei privilegi l'utente deve invocare il comando:

SET ROLE *NomeRuolo*;

In ogni istante un utente dispone dei privilegi che gli sono stati attribuiti direttamente e dei privilegi associati ai ruoli che ha attivato esplicitamente.

3.11 LE TRANSAZIONI

Una **transazione** è una unità elementare di lavoro svolta da una applicazione: consiste in una serie di operazioni che devono essere eseguite come se si trattasse di un'unica operazione.

Tutte le operazioni che compongono la transazione devono essere eseguite correttamente oppure non ne deve essere eseguita nessuna.
Se una delle operazioni fallisce tutte le operazioni precedentemente effettuate vengono annullate.
Per poter garantire questo comportamento le operazioni di una transazione non hanno alcun effetto sul database fino a quando la transazione non viene completata con successo.
Durante l'esecuzione di una transazione tutte le modifiche apportate sono considerate soltanto temporanee; se tutte le operazioni della transazione vengono eseguite con successo (**commit**) le modifiche apportate vengono registrate nel database, altrimenti tutte le modifiche vengono eliminate (**rollback**) e il database rimane nello stato in cui era prima dell'inizio della transazione.

La gestione delle transazioni permette di gestire l'**accesso concorrente** al database impedendo il verificarsi di situazioni di inconsistenza dei dati.

Generalmente si può realizzare una transazione racchiudendo le istruzioni tra i comandi **BEGIN TRANSACTION** e **END TRANSACTION**; all'interno possono comparire i comandi **COMMIT WORK** e **ROLLBACK WORK**.
La transazione ha effetto solo se viene eseguito il comando COMMIT.

Loescher Editore - Vietata la vendita e la diffusione

```
BEGIN TRANSACTION;
INSERT ...
UPDATE ...
COMMIT WORK;
END TRANSACTION;
```

Nota

Le istruzioni per la gestione delle transazioni dipendono dal DBMS utilizzato. Un esempio di transazioni è proposto nel capitolo di MySQL.

Proprietà ACID delle transazioni

Una transazione gode di particolari proprietà, chiamate acide dall'acronimo **ACID** (Atomicity, Consistency, Isolation, Durability):

- **Atomicity** (**atomicità**): rappresenta il fatto che una transazione è un'unità di esecuzione indivisibile (o atomica), cioè che viene eseguita tutta o per niente;
- **Consistency** (**consistenza** o **coerenza**): richiede che l'esecuzione della transazione non causi la violazione di vincoli definiti sul database; se ci sono violazioni non risolvibili al momento del commit la transazione viene annullata (rollback);
- **Isolation** (**isolamento**): richiede che l'esecuzione di una transazione sia indipendente dall'esecuzione contemporanea di altre transazioni: il risultato deve essere lo stesso che si avrebbe se la transazione venisse eseguita da sola;
- **Durability** (**persistenza**): richiede che l'effetto di una transazione che ha eseguito il commit sia duraturo e non possa essere perso; una transazione viene considerata conclusa solo se le modifiche sono state registrate nei file di log ed è quindi sempre possibile ripristinare la situazione anche in seguito ad un evento disastroso.

VERIFICA

Questionario

1. Che cos'è SQL?
2. Quali sono i principali standard relativi a SQL?
3. Quali sono i comandi SQL per la definizione dello schema?
4. Quali sono i comandi SQL per la gestione degli indici?
5. Quali sono i comandi SQL per la manipolazione dei dati?
6. Quali clausole si possono usare nel comando SELECT?
7. Quali sono i comandi SQL per la gestione delle viste?
8. Che cosa sono i trigger e quale paradigma seguono?
9. Che cosa sono le stored procedure?
10. Quali sono i comandi SQL per la gestione dei diritti di accesso?
11. Che cos'è una transazione?
12. Quali sono le proprietà ACID delle transazioni?

Loescher Editore - Vietata la vendita e la diffusione

Test

1 ..... *Indicare se le seguenti affermazioni sono vere o false.*

V	F	
☐	☐	1) Le tabelle temporanee sono calcolate dinamicamente.
☐	☐	2) Le modifiche ai dati del database influenzano anche le viste associate.
☐	☐	3) I trigger possono essere richiamati mediante il nome associato.
☐	☐	4) È possibile usare i trigger per creare file di log delle operazioni effettuate.
☐	☐	5) È possibile passare dei parametri alle stored procedure.
☐	☐	6) Si possono assegnare diritti di accesso alle viste.
☐	☐	7) Le modifiche apportate durante una transazione diventano effettive solo quando si esegue il comando COMMIT.
☐	☐	8) L'operazione di ROLLBACK consiste nell'annullamento dell'ultima operazione effettuata.
☐	☐	9) Una transazione è atomica se comprende un solo comando.
☐	☐	10) La proprietà di isolamento di una transazione indica che il risultato della transazione è lo stesso sia che la transazione venga eseguita da sola, sia che venga eseguita insieme ad altre transazioni.

2 ..... *Dire se le seguenti query relative alla relazione*
Oggetti (IdOggetto, Colore, Forma, Pezzi)
rappresentano una proiezione, una selezione o entrambe.

Proiezione	Selezione	
☐	☐	1) SELECT * FROM Oggetti;
☐	☐	2) SELECT * FROM Oggetti WHERE Pezzi > 10;
☐	☐	3) SELECT Colore, Forma FROM Oggetti WHERE Pezzi > 10;
☐	☐	4) SELECT Colore, Forma, Pezzi FROM Oggetti;
☐	☐	5) SELECT IdOggetto, Colore, Forma, Pezzi FROM Oggetti;
☐	☐	6) SELECT IdOggetto, Colore, Forma, Pezzi FROM Oggetti WHERE Colore='Rosso';

Esercizi

1 ..... Dopo aver individuato i tipi di dati degli attributi dello schema di riferimento di p. 39:
- scrivere i comandi SQL per creare le relazioni;
- scrivere i comandi SQL per popolare il database;
- scrivere i comandi SQL per visualizzare il contenuto di ciascuna tabella;
- scrivere la query SQL per individuare i fiumi che attraversano più nazioni e per ciascun fiume visualizzare la lunghezza del fiume, le nazioni che attraversa e il continente a cui appartengono.

Loescher Editore - Vietata la vendita e la diffusione

2 ..... Relativamente agli esercizi 3, 4 e 5 del capitolo 2:
- scrivere i comandi SQL per creare le relazioni;
- scrivere i comandi SQL per popolare il database;
- scrivere i comandi SQL corrispondenti alle espressioni dell'algebra relazionale individuate.

3 ..... Per lo schema relazionale dell'esercizio 3 del capitolo 2 scrivere le query SQL per:
1) visualizzare per ogni studente: matricola, cognome ed elenco degli esami sostenuti e non superati (voto < 18);
2) visualizzare per ogni studente: matricola, cognome, numero degli esami superati e media dei voti riportati;
3) elencare cognome e nome degli studenti che hanno superato tutti gli esami sostenuti;
4) individuare le materie per cui non sono stati sostenuti esami nell'anno corrente;
5) individuare lo studente che ha la media più alta;
6) elencare le materie in cui ci sono studenti che non hanno ancora superato l'esame e il numero di tali studenti;
7) individuare lo studente che ha ripetuto uno stesso esame il maggior numero di volte, specificando la materia in cui ciò si è verificato;
8) individuare la materia che ha la percentuale maggiore di risultati insufficienti;
9) visualizzare l'elenco delle materie e per ogni anno la media dei voti assegnati;
10) visualizzare cognome e nome degli studenti che hanno riportato per tutti gli esami voti superiori alla media, indicando l'elenco degli esami col voto e lo scarto rispetto alla media;
11) individuare il voto medio complessivo di tutti gli esami sostenuti ed elencare le materie per cui il numero dei voti assegnati superiori alla media è maggiore dei voti assegnati inferiori alla media.

4 ..... Per lo schema relazionale dell'esercizio 4 del capitolo 2 scrivere le query SQL per:
1) visualizzare nome e cognome delle persone che si sono sposate prima di compiere 25 anni;
2) visualizzare nome e cognome degli uomini più giovani delle loro mogli;
3) individuare gli uomini che si sono sposati più volte, fornendo nome e cognome di ciascuna moglie;
4) individuare le donne che si sono sposate più volte, fornendo nome e cognome dell'ultimo marito;
5) individuare la città in cui si sono svolti più matrimoni;
6) elencare per ogni anno il numero di matrimoni;
7) calcolare la percentuale dei matrimoni in cui uno dei coniugi si è sposato più di una volta rispetto al numero dei matrimoni.

PROPOSTE DI LAVORO

1 ..... Dato lo schema relazionale
Materie (<u>IdMateria</u>, DescrizioneM)
Classi (<u>NomeC</u>, Aula, Rappresentante)
Studenti (<u>Matricola</u>, Cognome, Nome, Classe)
Pagelle (<u>Studente</u>, <u>Materia</u>, Voto)
Studenti.Classe è in integrità referenziale con *Classi.NomeC*;
Classi.Rappresentante è in integrità referenziale con *Studenti.Matricola*;
Pagelle.Studente è in integrità referenziale con *Studenti.Matricola*;
Pagelle.Materia è in integrità referenziale con *Materie.IdMateria*;

Loescher Editore - Vietata la vendita e la diffusione

- creare le tabelle:

1) *Materie* con chiave *IdMateria* numerico di due cifre e *DescrizioneM* di 20 caratteri; nessuno dei due campi può essere NULL;
2) *Classi* con chiave *NomeC* di due caratteri (una cifra che indica la classe e una lettera che indica la sezione), *Aula* numerico di tre cifre che indica il numero dell'aula occupata e può avere il valore NULL se non è noto ma, se presente, deve essere univoco, *Rappresentante* numerico di cinque cifre, chiave esterna in integrità referenziale con *Studenti.Matricola* (può essere NULL se il rappresentante non è noto);
3) *Studenti* con chiave *Matricola* numerico di quattro cifre che rappresenta un numero progressivo che identifica lo studente, *Cognome* e *Nome* di 20 caratteri ciascuno, *Classe* di due caratteri, chiave esterna in integrità referenziale con *Classi.NomeC* che rappresenta la classe a cui appartiene lo studente per l'anno in corso e non può essere NULL;
4) *Pagelle* con chiave composta da *Studente* e *Materia*; *Studente* è un numero di quattro cifre chiave esterna in integrità referenziale con *Studenti.Matricola* e *Materia* è un numero di due cifre in integrità referenziale con *Materie.IdMateria*; *Voto* deve essere compreso tra 1 e 10;

- creare gli indici:

1) sull'attributo *Classe* di *Studente*;
2) sugli attributi *Materia* e *Studente* di *Pagelle*;

- effettuare le seguenti operazioni:

1) inserire nella tabella *Materie* i dati: 1 - Italiano, 2 - Matematica, 3 - Informatica, 4 - Elettronica, 5 - Inglese;
2) inserire nella tabella *Classi* i dati relativi alle classi III, IV e V delle sezioni A e B; la classe 3A è assegnata all'aula 101, la classe 4A all'aula 102, la classe 3 B all'aula 201; i rappresentanti per il momento non sono noti;
3) inserire nella tabella *Studenti* alcuni dati, assegnando alcuni studenti a ogni classe; in particolare inserire uno studente con matricola 1500 appartenente alla classe 3B;
4) modificare la tabella *Classi* aggiungendo alle classi della sezione B la matricola del rappresentante; per la classe 3B assegnare come rappresentante lo studente con matricola 1500;
5) inserire nella tabella *Voti* alcuni dati;
6) cancellare dalla tabella *Voti* tutti i voti dello studente con matricola 1500;
7) cancellare dalla tabella *Studenti* lo studente con matricola 1500 che si è ritirato; che cosa deve accadere nella tabella classi alla riga relativa alla 3B dato che 1500 era indicato come rappresentante?
8) aumentare di 1 nelle pagelle tutti i voti che sono inferiori a 3;

- scrivere le query SQL per individuare:

1) l'elenco delle materie, codice e descrizione;
2) l'elenco di tutte le classi e l'aula che occupano;
3) l'elenco delle descrizioni delle materie con intestazione 'Materie';
4) gli studenti della 5A e inserirli in una tabella temporanea;
5) l'elenco delle classi di cui non è noto il rappresentante;
6) l'elenco delle classi che occupano le aule comprese tra la 101 e la 110;
7) l'elenco degli studenti il cui cognome comincia per S;
8) l'elenco degli studenti di tutte le classi della sezione B;
9) l'elenco delle materie in ordine alfabetico;
10) l'elenco degli studenti divisi per classe e in ordine alfabetico per ciascuna classe;
11) il numero degli studenti della classe 4A;
12) la media dei voti delle pagelle di tutti gli studenti;

Loescher Editore - Vietata la vendita e la diffusione

13) la media dei voti dello studente con matricola 1600;
14) i voti più basso e più alto che sono stati assegnati in pagella;
15) il numero di studenti di ciascuna classe;
16) le pagelle degli studenti con matricola dello studente, descrizione della materia e voto;
17) le pagelle degli studenti con cognome e nome dello studente, descrizione della materia e voto;
18) i nomi degli studenti che hanno l'insufficienza in inglese;
19) per ogni studente nome, cognome e media dei voti;
20) le materie in cui sono stati assegnati voti insufficienti;
21) il nome dello studente che ha il voto più alto in matematica;
22) la materia che ha il più alto numero di insufficienze;
23) i nomi degli studenti che hanno almeno una insufficienza;
24) i nomi degli studenti che hanno tutti i voti sufficienti;
25) i nomi degli studenti della 5B che hanno la media superiore a quella di tutti gli studenti della 5A;

- creare una vista con i voti di matematica che comprenda classe, cognome e nome dello studente e voto in matematica, poi, usando la vista, scrivere le query SQL per individuare:

1) la percentuale di insufficienze in matematica;
2) la media dei voti di matematica per ogni classe;
3) lo studente di ogni classe che ha il voto più alto in matematica;
4) la classe in cui la percentuale di insufficienze è più alta.

2 ..... Dopo aver creato e popolato le tabelle dello schema relazionale
Nazioni (NomeN, Continente, Capitale)
Olimpiadi (Anno, Nazione, Città)
Olimpiadi.Nazione è integrità referenziale con *Nazioni.NomeN*;
scrivere le query SQL per individuare:
1) l'elenco ordinato per anno delle olimpiadi con città e nazione in cui si sono svolte;
2) il nome delle nazioni che hanno ospitato almeno una olimpiade;
3) il nome delle nazioni che hanno ospitato più di una olimpiade;
4) il nome della nazioni che hanno ospitato le olimpiadi in una città diversa dalla capitale;
5) il continente che ha ospitato più olimpiadi.

3 ..... Dopo aver creato e popolato le tabelle dello schema relazionale
CD (CodiceCD, Autore, Titolo, Durata)
Clienti (CodiceCliente, Nome, Citta)
Affitto (CD, Cliente, DataAffitto, DataRestituzione)
Affitto.DataRestituzione ha valore NULL se il CD non è ancora stato restituito;
Affitto.CD è in integrità referenziale con *CD.CodiceCD*;
Affitto.Cliente è in integrità referenziale con *Clienti.CodiceCliente*;
scrivere le query SQL per individuare:
1) il codice del CD di durata più lunga;
2) per ogni autore il numero dei CD in catalogo;
3) gli autori di cui almeno un CD è in affitto;
4) l'elenco dei CD in affitto con nome del cliente e data di affitto;
5) il CD che è stato affittato più volte, indicandone anche l'autore e il titolo;
6) l'elenco dei clienti suddivisi per città e in ordine alfabetico per ogni città;
7) per ogni autore il numero dei CD in affitto;
8) per ogni autore l'elenco delle città in cui abitano i clienti che hanno un CD in affitto;
9) il nome del cliente che ha tenuto un CD per il tempo più lungo e il titolo di tale CD;
10) il nome dei clienti che hanno tenuto almeno un CD per un tempo superiore alla media;

Loescher Editore - Vietata la vendita e la diffusione

11) la città i cui sono stati affittati più CD;
12) il nome dei clienti che hanno preso in prestito almeno 2 CD diversi di Lucio Battisti;
13) il numero di CD presi in prestito di ogni autore, ordinando il risultato in maniera decrescente per numero di CD;
14) il nome dei clienti che hanno preso in prestito tutti i CD dei Beatles.

4 ..... Dopo aver creato e popolato le tabelle dello schema relazionale
Film (Titolo, CodRegista, Anno, Genere)
AttReg (Codice, Nome, Attore, Regista) in cui *Regista* e *Attore* sono campi booleani
Recitazione (Attore, Film)
Film.CodRegista è in integrità referenziale con *AttReg.Codice*;
Recitazione.Attore è in integrità referenziale con *AttReg.Codice*;
Recitazione.Film è in integrità referenziale con *Film.Titolo*;
scrivere le query SQL per individuare:
1) i nomi delle persone che sono sia attori che registi;
2) l'elenco dei film suddivisi per genere e per ogni genere in ordine di anno;
3) l'elenco dei film in cui ha recitato Robert De Niro;
4) l'elenco dei film e dei relativi registi in cui hanno recitato sia Robert De Niro che Al Pacino;
5) l'elenco dei film in cui il regista è stato anche attore del film;
6) per ogni regista l'elenco dei film che ha diretto e degli attori che vi hanno recitato;
7) i nomi degli attori che hanno recitato in film sempre dello stesso genere;
8) i nomi dei registi che hanno diretto film di almeno due generi diversi;
9) i nomi degli attori che hanno recitato solo in film diretti da loro stessi;
10) i nomi degli attori che hanno recitato in almeno un film diretto da Woody Allen;
11) i nomi degli attori che hanno recitato in tutti i film diretti da Woody Allen;
12) le coppie di attori che hanno recitato esattamente negli stessi film.

5 ..... Dopo aver creato e popolato le tabelle dello schema relazionale
Citta (IdCitta, NomeC, Nazione)
Musei (IdMuseo, NomeM, Citta)
Opere (IdOpera, Titolo, Tipo, Anno, Museo, Autore) con *Tipo* = {'pittura', 'scultura', ...}
Artisti (IdArtista, Nome, Nazione, AnnoNascita, AnnoMorte)
Musei.Citta è in integrità referenziale con *Citta.IdCitta*;
Opere.Museo è in integrità referenziale con *Musei.IdMuseo*;
Opere.Autore è in integrità referenziale con *Artisti.IdArtista*;
scrivere le query SQL per individuare:
1) l'elenco dei musei italiani raggruppati per città, in ordine alfabetico di città e per ogni città in ordine alfabetico per nome;
2) l'elenco delle opere che si trovano al Louvre con titolo dell'opera e nome dell'autore;
3) l'elenco delle opere di Leonardo con il titolo e il museo in cui si trovano (nome, del museo, città e nazione);
4) i nomi delle nazioni in cui si trovano opere di Picasso;
5) i nomi degli autori le cui opere si trovano in un solo museo;
6) il nome dell'autore le cui opere sono divise nel più alto numero di musei;
7) l'elenco delle opere che sono state create dall'artista nell'anno in cui è morto;
8) i nomi degli artisti le cui opere si trovano tutte in musei della stessa nazione in cui è nato;
9) per ogni artista l'elenco dei tipi di opere che ha realizzato;
10) per ogni museo il titolo dell'opera più antica e dell'opera più recente custodite;
11) l'elenco dei musei in cui sono custodite opere di artisti italiani e per ciascun museo il numero di tali opere.

Loescher Editore - Vietata la vendita e la diffusione

Modulo 2

La progettazione di database

4 Il modello concettuale

5 Il modello logico

6 Il modello fisico

Loescher Editore - Vietata la vendita e la diffusione

4 Il modello concettuale

Prerequisiti

- Terminologia generale relativa ai database
- Fasi della progettazione di un database

Obiettivi

Conoscenze

- Modello ER e UML
- Costrutti del modello concettuale: entità e attributi, identificatori interni ed esterni, associazioni, generalizzazioni
- Regole aziendali e vincoli di integrità
- Principi di modellazione: scelta fra entità e attributi e fra associazioni e entità

Abilità/capacità

- Modellare frammenti di realtà con i costrutti del modello concettuale
- Verificare un modello concettuale mediante le regole di lettura
- Modellare gli attributi multivalore come entità
- Scegliere fra l'uso di associazioni e di entità

Loescher Editore - Vietata la vendita e la diffusione

4.1 MODELLO ER E UML

Lo **schema concettuale** di un database è una rappresentazione della realtà di interesse secondo un determinato modello basato su associazioni tra entità.

Nello schema concettuale vanno rappresentati solo entità, attributi e associazioni di interesse per l'applicazione.

Nota

Per descrivere l'organizzazione e la struttura dei dati si usano dei **costrutti** che definiscono gli schemi di base, da combinare per descrivere realtà complesse.
I costrutti principali permettono di rappresentare entità, attributi e associazioni.

I costrutti possono essere descritti tramite due tipi di **rappresentazioni grafiche**:

- diagrammi entità/associazioni (o diagrammi ER);
- diagrammi UML.

Il **modello entità/associazioni** o **modello ER** (Entity Relationship) è stato introdotto nel 1976 da Peter P. Chan appositamente come modello grafico per la descrizione dei dati e delle loro associazioni.

Il modello ER è chiamato a volte modello entità/relazioni come sinonimo di entità/associazioni, ma dato che il termine relazione nel modello relazionale ha un significato diverso verrà sempre usato il termine associazione.

Nota

UML (Unified Modelling Language - linguaggio unificato di modellazione) è stato progettato a metà degli anni 90 ed è stato standardizzato dall'Object Management Group (OMG).

OMG è un consorzio industriale che si occupa di favorire l'interoperabilità tra applicazioni software sviluppate da aziende diverse attraverso la definizione di specifiche concordate dai suoi membri.

Nota

UML è uno standard per la progettazione nella programmazione orientata agli oggetti, ma si sta diffondendo nella progettazione di tutti i sistemi software, anche i database.
Per rappresentare dati e associazioni vengono usati i diagrammi delle classi, infatti i concetti del modello ER sono riconducibili a quelli relativi ai diagrammi di classe.

Confronto tra il modello ER e UML

Il modello ER è stato ideato appositamente per la rappresentazione concettuale dei dati; i diagrammi UML invece sono stati ideati per la progettazione di applicazioni con il paradigma a oggetti: alcuni costrutti rilevanti per la modellazione dei dati non sono previsti in UML e si adottano quindi delle notazioni non standard.
UML però permette di rappresentare anche tutti gli altri aspetti di una applicazione software.

Principali diagrammi UML

- **Diagrammi di interazione** tra oggetti: descrivono le chiamate di metodi (cioè lo scambio di messaggi) in un caso specifico. Ci sono due forme di diagrammi di interazione tra oggetti: il **diagramma di collaborazione** e il **diagramma di sequenza**; i due diagrammi possono essere convertiti uno nell'altro. Nel **diagramma di collaborazione** vengono

Loescher Editore - Vietata la vendita e la diffusione

rappresentate le chiamate di metodi con i parametri. Il **diagramma di sequenza** mostra la sequenza temporale delle operazioni. Si può vedere non solo a chi appartengono i metodi richiamati ma anche quando vengono richiamati (cioè non solo a chi sono diretti i messaggi ma anche quando vengono inviati).

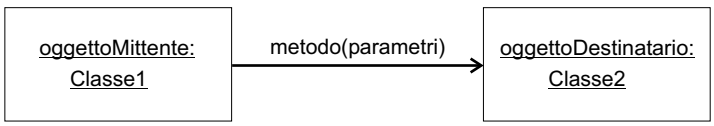


Figura 4.1 *Diagramma di collaborazione*

- **Diagrammi di interfaccia**: servono per descrivere l'interfaccia utente. Il **diagramma della disposizione interna delle finestre** serve per descrivere l'organizzazione delle finestre: la posizione dei campi, caratteristiche come campi obbligatori e facoltativi, ricerche in database e così via. Il **diagramma di navigazione** tra le finestre indica in che modo si può passare da una finestra all'altra.

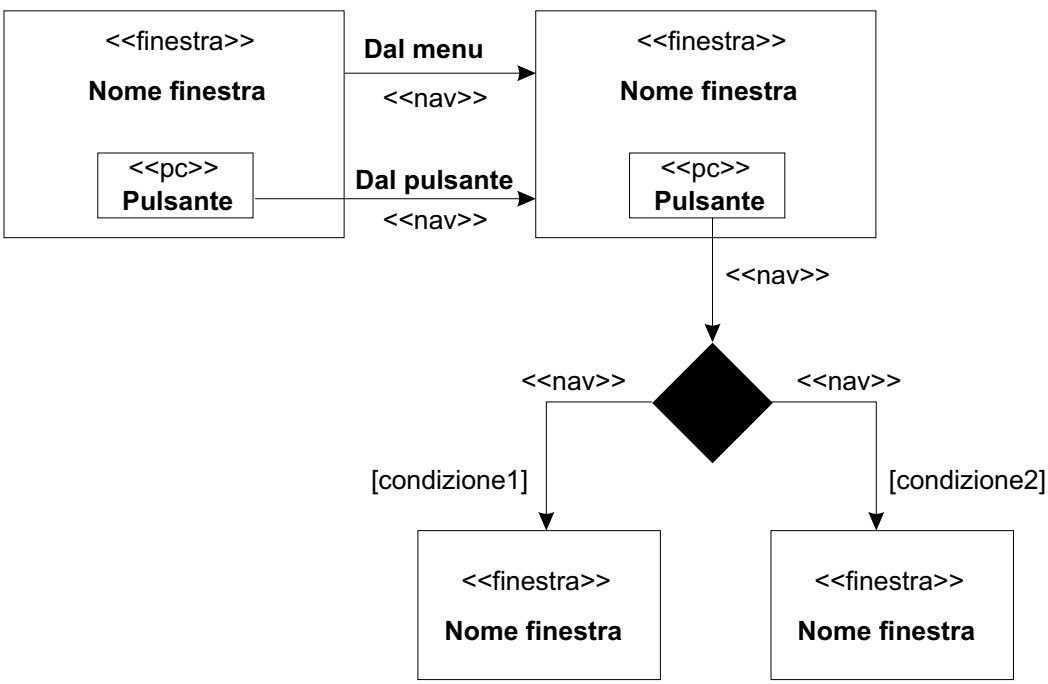
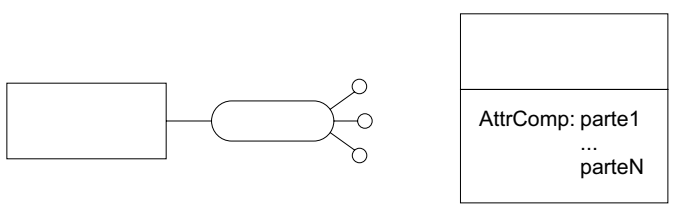
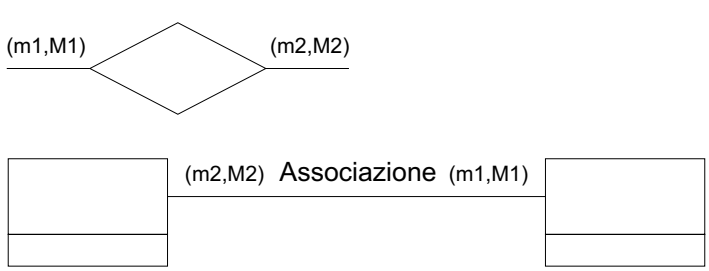
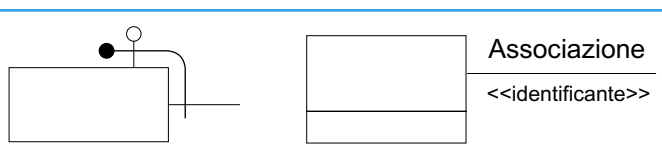


Figura 4.2 *Diagramma di navigazione*

- **Diagramma dei casi d'uso**: descrive le modalità di utilizzo del sistema da parte degli attori (persone o sistemi che interagiscono con il sistema) e le modalità di interazione tra attori e sistema.
- **Diagramma delle attività**: descrive il comportamento dinamico di un processo dell'applicazione attraverso flussi di attività da svolgere.
- **Diagramma degli stati**: descrive il ciclo di vita di un oggetto dell'applicazione attraverso gli stati che può assumere.
- **Diagrammi dei componenti** e **di distribuzione dei componenti**: rappresentano l'organizzazione dei componenti software del sistema, la disposizione dei nodi hardware e le loro dipendenze.

Loescher Editore - Vietata la vendita e la diffusione

Costrutti principali in ER e UML

	Entità
	Attributo semplice con eventuale cardinalità (molteplicità)
	Attributo composto
	Chiave
	Chiave composta
	Associazione con cardinalità (molteplicità)
	Generalizzazione
	Identificatore esterno

Loescher Editore - Vietata la vendita e la diffusione

4.2 ENTITÀ

Una **entità** rappresenta una classe di oggetti con proprietà comuni e esistenza autonoma.

Una **istanza** di una entità è un esemplare, cioè un oggetto della classe che l'entità rappresenta.

Nota

A volte le istanze vengono chiamate entità, mentre le entità vengono chiamate tipi di entità o categorie.

Entità ***Nazione***: classe delle nazioni; ogni nazione è un esemplare o istanza dell'entità *Nazione*.
Entità ***Persona***: classe delle persone; ogni persona è un esemplare o istanza dell'entità *Persona*.

L'insieme delle istanze (**estensione**) può cambiare nel tempo e non è l'insieme di tutte le possibili istanze di quel tipo, ma solo di quelle che interessano in un certo contesto in un certo momento.

Nazione: solo quelle dell'area geografica considerata.
Persona: in una classe gli studenti della classe.

Ogni entità ha un nome che la identifica univocamente; come **nomi delle entità** si usano sostantivi, per convenzione al singolare con iniziale maiuscola (se si usano nomi composti anche ogni parola successiva ha iniziale maiuscola).

Nel **modello ER** ogni entità viene rappresentata con un rettangolo con il nome dell'entità all'interno.

In **UML** ogni entità viene rappresentata con un rettangolo con il nome dell'entità all'interno; può esserci una parte inferiore separata.

Figura 4.3
Rappresentazione delle entità

Nota

La parte inferiore è predisposta per contenere la lista degli attributi.

Ogni entità viene descritta mediante **proprietà** che la caratterizzano.

Entità	Proprietà
Nazione	nome, continente, capitale, superficie, popolazione, bandiera, inno, sigla, tipo di governo, regioni interne ecc.
Persona	nome, cognome, codice fiscale, sesso, data di nascita, indirizzo, numero di telefono, hobby ecc.

Nota

Le proprietà sono rappresentate negli schemi dal costrutto attributi.

ENTITÀ

Una entità è un elemento concreto o astratto dotato di certe proprietà descritte da attributi.

Loescher Editore - Vietata la vendita e la diffusione

4.3 ATTRIBUTI

Gli **attributi** descrivono le proprietà elementari di entità.

Come **nomi degli attributi** si usano sostantivi, per convenzione al singolare con iniziale maiuscola (se si usano nomi composti anche ogni parola successiva ha iniziale maiuscola).

Nel **modello ER** gli attributi vengono rappresentati con cerchi collegati da segmenti alle entità.

In **UML** gli attributi sono elencati nella parte inferiore del rettangolo che rappresenta l'entità con una linea di separazione tra il nome dell'entità e la lista degli attributi.

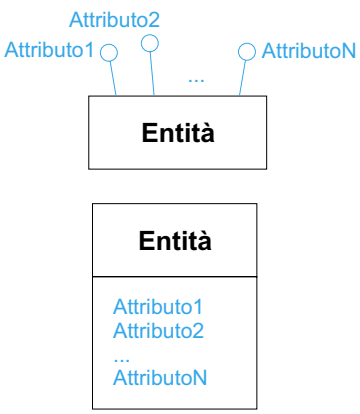


Figura 4.4
Rappresentazione degli attributi

Dominio degli attributi

Un attributo associa un **valore** a ciascuna istanza; i valori assunti dagli attributi caratterizzano e determinano le istanze delle entità.

Il valore può essere **costante** o variabile; gli attributi costanti non possono essere modificati dopo che l'entità è stata creata, quelli variabili sì.

> ***Persona***: la data di nascita è un attributo costante, l'indirizzo o il numero di telefono sono attributi variabili.

I valori ammissibili per ogni attributo sono solo quelli di un insieme detto **dominio** dell'attributo.

Il **dominio** può essere identificato da:

- **tipo** di valori che può assumere (per esempio numerico o stringa);

  In base al **tipo** di dati gli attributi si distinguono in:

  - **semplici** o **atomici** se il dominio è un tipo di dato semplice;
  - **composti** o **aggregati** se sono di tipo record; gli attributi si possono raggruppare in attributi composti in modo ricorsivo.

  > Il tipo di un attributo non può essere una entità (per esempio il coniuge di una persona non può essere ancora di tipo *Persona*). **Nota**

  Immagini e suoni possono essere rappresentati con i riferimenti ai file che li contengono: path, nome (eventualmente autoesplicativo) ed estensione.

  > ***Nazione***: la bandiera (immagine) e l'inno (file audio) sono rappresentati dai riferimenti ai relativi file, che possono avere come nome il nome della nazione stessa.

- **dimensione**: numero massimo di cifre o di caratteri utilizzabili;
- **vincoli**: limitazioni sui valori o intervalli di valori ammessi.

Loescher Editore - Vietata la vendita e la diffusione

Si dice **attributo enumerativo** (o a volte anche attributo unione) un attributo che può assumere un valore tra un insieme finito di possibilità.

I domini non vengono rappresentati nello schema grafico ma vengono descritti nella **documentazione** associata.

Cardinalità degli attributi

La **cardinalità degli attributi** descrive il numero minimo e massimo di valori dell'attributo associati a ogni istanza di entità.

Di solito la cardinalità è (1, 1) e viene omessa; a volte però il valore di un attributo può essere nullo (minima 0) o avere più valori (massima N).

In base alla **cardinalità** gli attributi si distinguono in:

- **obbligatori** se il valore deve essere sempre presente;
- opzionali o **facoltativi** se il valore per una certa entità può essere nullo;

Il valore nullo (***null***) rappresenta una informazione mancante perché inapplicabile o sconosciuta.

- **multivalore** se possono essere presenti più valori per un attributo.

Significato della cardinalità

minima 0: attributo opzionale;
minima 1: attributo obbligatorio;
massima N: attributo multivalore.

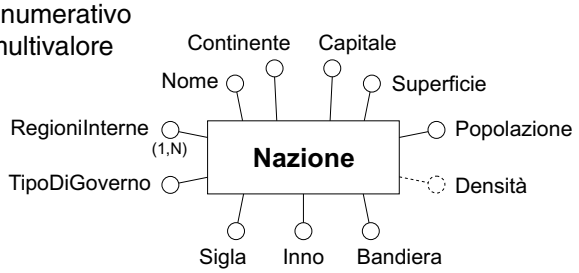
Attributi derivati

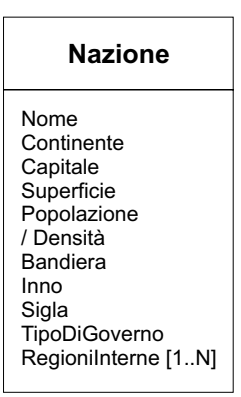
Si dicono **attributi derivati** quelli che sono ottenibili dal calcolo di una espressione costruita con attributi dell'entità.

Esempio 4.1

Nazione

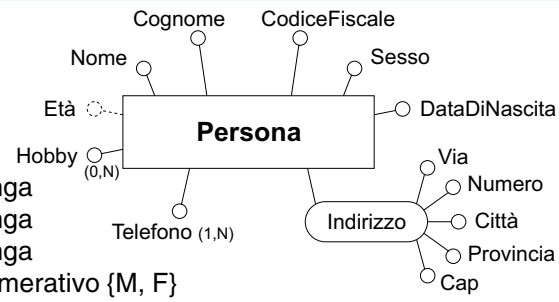
Nome	stringa
Continente	enumerativo {Europa, Asia, Africa, America, Oceania}
Capitale	stringa
Superficie	numerico
Popolazione	numerico
Densità	derivato *Popolazione / Superficie*
Bandiera	immagine
Inno	suono
Sigla	stringa
TipoDiGoverno	enumerativo
RegioniInterne	multivalore





Loescher Editore - Vietata la vendita e la diffusione

Esempio 4.2



Persona	
Nome	stringa
Cognome	stringa
CodiceFiscale	stringa
Sesso	enumerativo {M, F}
DataDiNascita	elementare o composto da giorno mese e anno
Indirizzo	composto da via, numero, città, provincia e CAP
Telefono	multivalore
Hobby	multivalore opzionale (può avere valore *null*)
Età	derivato *annoCorrente - AnnoDiNascita*

Persona

Nome
Cognome
CodiceFiscale
Sesso
DataDiNascita
Indirizzo: Via
Numero
Città
Provincia
Cap
Telefono [1..N]
Hobby [0..N]
/ Età

Identificatori e chiavi

Si dice **identificatore** di una entità un concetto che permette di identificare in modo univoco le istanze dell'entità.

L'identificatore normalmente è costituito da uno o più attributi dell'entità (**identificatore interno**).

L'identificatore a volte può essere esterno, cioè una entità è identificata mediante un'altra entità. **Nota**

Ogni entità può avere anche più di un identificatore; si dice **chiave candidata** ciascun identificatore interno.

Le chiavi candidate a volte sono chiamate semplicemente chiavi. **Nota**

Chiave candidata

Una **chiave candidata** per un'entità è un insieme minimale di uno o più attributi che permettono di distinguere univocamente le istanze dell'entità.
La chiave cioè deve avere un valore diverso per ogni entità e togliendo uno degli attributi ciò non è più vero.

Tra le chiavi candidate ne viene scelta una come **chiave primaria** (PK - Primary Key), di solito quella con il minor numero di attributi.

Ogni attributo di una chiave composta da più attributi è un PPK (Partial Primary Key). **Nota**

Quando si parla genericamente di chiave di una entità si intende di solito la chiave primaria. **Nota**

A volte la chiave può essere definita in modo artificiale, introducendo un **attributo fittizio**: un **codice** o un **numero progressivo** (che può essere assegnato automaticamente).

L'introduzione di una chiave artificiale è utile quando l'entità non ha una chiave primaria (**entità debole**) o per motivi di efficienza quando tutte le chiavi candidate hanno molti attributi o comunque sono molto lunghe.

Loescher Editore - Vietata la vendita e la diffusione

Nel **modello ER** l'attributo chiave viene evidenziato sottolineandolo o riempiendo il cerchio con il colore nero.
Se la chiave è composta da più attributi, gli attributi vengono collegati insieme con un segmento con all'estremità un cerchio nero.

In **UML** vicino alla chiave primaria si indica {PK} (primary Key); se è composta da più attributi per ognuno si indica {PPK} (Partial Primary Key).

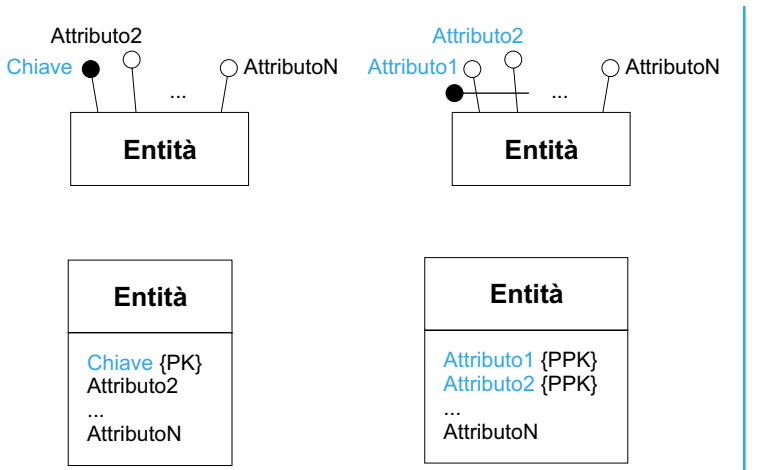
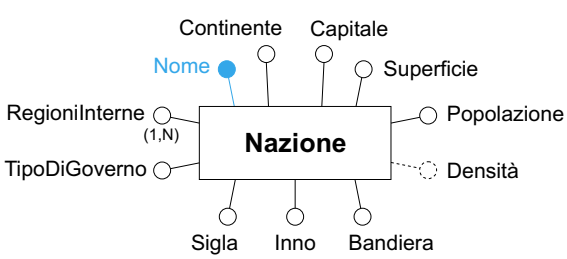


Figura 4.5 *Rappresentazione della chiave*

Esempio 4.3

Chiavi alternative per *Nazione*: *Nome* o *Sigla*.

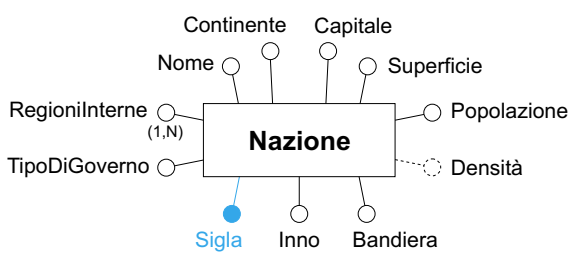


Nazione

Nome {PK}
Continente
Capitale
Superficie
Popolazione
/ Densità
Bandiera
Inno
Sigla
TipoDiGoverno
RegioniInterne [1..N]

Nazione

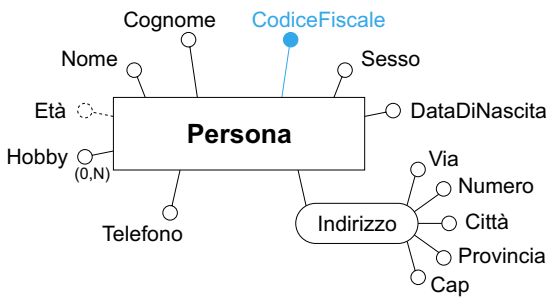
Nome
Continente
Capitale
Superficie
Popolazione
/ Densità
Bandiera
Inno
Sigla {PK}
TipoDiGoverno
RegioniInterne [1..N]



Esempio 4.4

Chiavi alternative per *Persona*: *CodiceFiscale* o {*Nome*, *DataDiNascita*, *Indirizzo*}.

Loescher Editore - Vietata la vendita e la diffusione

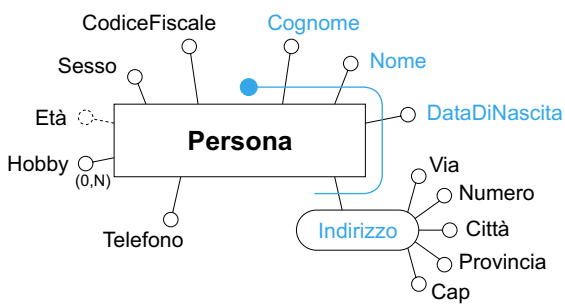


Persona

Nome
Cognome
CodiceFiscale {PK}
Sesso
DataDiNascita
Indirizzo: Via
Numero
Città
Provincia
Cap
Telefono [1..N]
Hobby [0..N]
/ Età

Persona

Nome {PPK}
Cognome {PPK}
CodiceFiscale
Sesso
DataDiNascita {PPK}
Indirizzo {PPK}: Via
Numero
Città
Provincia
Cap
Telefono [1..N]
Hobby [0..N]
/ Età



4.4 ASSOCIAZIONI

Un'**associazione** è un legame logico tra entità.

Le associazioni sono a volte dette anche relazioni dal termine inglese relationship ma è preferibile non usare questo termine per non fare confusione con le relazioni del modello relazionale.

Nota

Una associazione ha due **versi**: è composta da due associazioni, una **diretta** e l'altra **inversa**. Le due associazioni si differenziano solo per l'ordine con cui vengono considerate le entità.

Ogni associazione è identificata da un nome (con iniziali maiuscole).
Il **nome dell'associazione** può essere:

- un sostantivo se non si vuole indurre a individuare un verso;
- un verbo per individuare un verso: si usano verbi attivi in un verso e passivi nell'altro.

Anche le associazioni possono avere **attributi**.

Si dice **grado** di una associazione il numero di entità che partecipano all'associazione.

Le associazioni sono per la maggior parte **binarie**, cioè tra due entità.
Ci possono essere associazioni tra più entità; si dice associazione **n-aria** un'associazione che coinvolge *n* entità.

Loescher Editore - Vietata la vendita e la diffusione

Nel **modello ER** ogni associazione viene rappresentata con un rombo con il nome dell'associazione all'interno, collegato alle entità in relazione.

Gli attributi dell'associazione vengono indicati con cerchi collegati da segmenti al rombo.

In **UML** ogni associazione viene rappresentate da una linea che unisce le entità in relazione; vicino alla linea viene indicato il nome; se per i nomi delle associazioni si usano verbi un triangolino indica il verso dell'associazione.

Gli attributi dell'associazione vanno indicati nella parte inferiore di un rettangolo come quello delle entità, collegato alla linea che rappresenta l'associazione da una linea tratteggiata.

Le associazioni ternarie in UML sono rappresentate da un rombo come in ER (collegando al rombo con una linea tratteggiata il rettangolo con le proprietà dell'associazione).

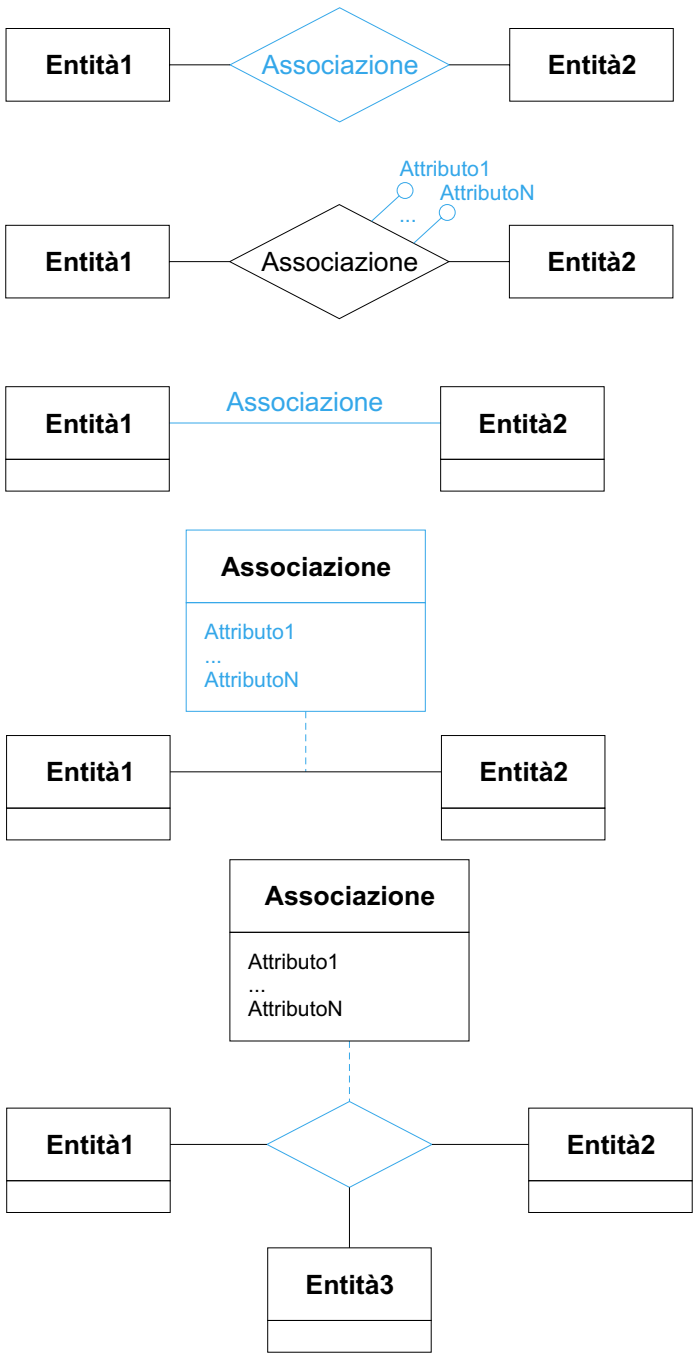
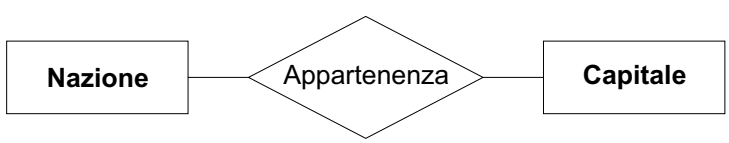


Figura 4.6
Rappresentazione delle associazioni

Esempio 4.5

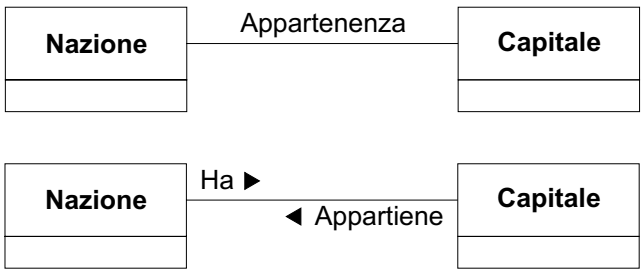
Associazione *Appartenenza* tra le entità *Nazione* e *Capitale*.



Loescher Editore - Vietata la vendita e la diffusione

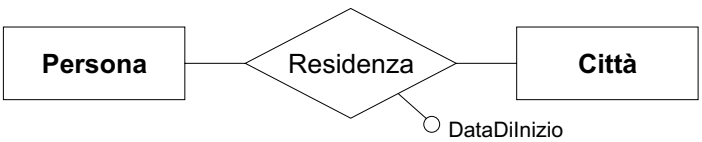
Si possono separare le associazioni diretta e inversa:

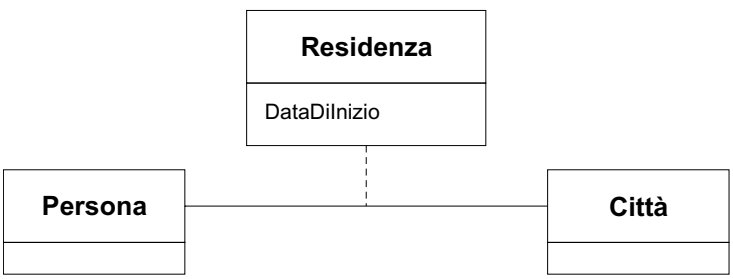
Una *Nazione Ha* una *Capitale.*
Una *Capitale Appartiene* a una *Nazione.*



Esempio 4.6

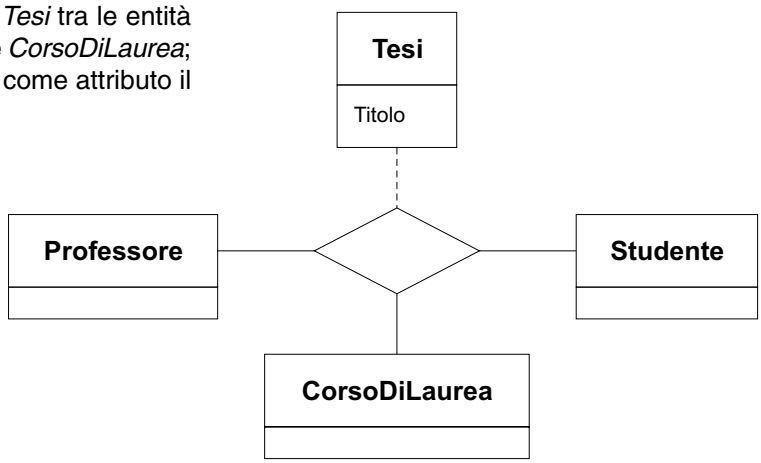
Associazione *Residenza* tra le entità *Persona* e *Città*; l'associazione *Residenza* ha come attributo la data di inizio.

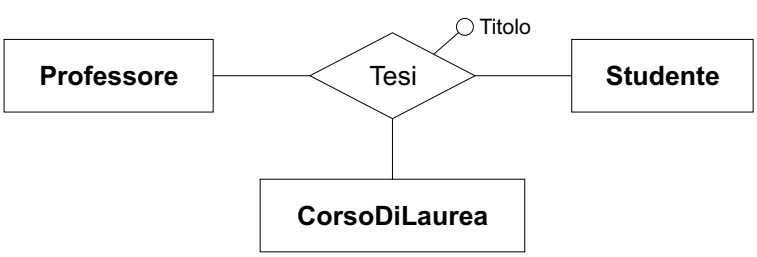




Esempio 4.7

Associazione ternaria *Tesi* tra le entità *Studente*, *Professore* e *CorsoDiLaurea*; l'associazione *Tesi* ha come attributo il *Titolo*.





Loescher Editore - Vietata la vendita e la diffusione

Una **istanza di associazione** è un insieme di istanze di entità, una istanza per ogni entità.
Per una associazione **binaria** una istanza è una **coppia di istanze di entità**.
Per una associazione n-aria una istanza è una ennupla di istanze di entità.

Definizione formale di associazione

L'insieme delle istanze di una associazione è una relazione matematica tra le istanze delle entità.

Una associazione su *n* entità E_1, E_2, ... E_n è un sottoinsieme del prodotto cartesiano E_1 x E_2 x ... x E_n, cioè, se la ennupla e_1, e_2, ... e_n, appartiene all'associazione, vuol dire che esiste un legame tra le istanze e_1, e_2, ... e_n.

Dato che una associazione è un insieme (sottoinsieme del prodotto cartesiano) non ci possono essere ennuple ripetute.

Cardinalità o molteplicità delle associazioni

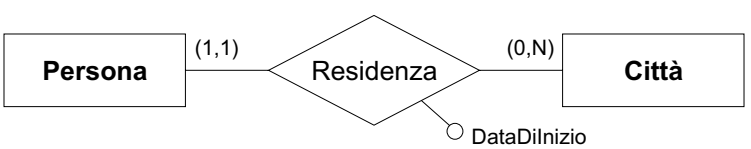
Cardinalità e **molteplicità** di una associazione sono concetti simili usati rispettivamente nel modello ER e in UML per indicare il numero di possibili collegamenti tra istanze di una entità e istanze delle altre entità coinvolte nell'associazione.
In realtà si utilizza una coppia di valori che indica rispettivamente il numero minimo e massimo di collegamenti possibili.
I valori utilizzati per la cardinalità/molteplicità sono 0, 1 e N (molti, indica genericamente un intero maggiore di 1, teoricamente senza un limite superiore).

Cardinalità nel modello ER

Nello schema ER le cardinalità vengono specificate per ciascuna partecipazione di entità a una associazione e indicano il numero minimo e massimo di istanze di associazione a cui una istanza dell'entità può partecipare.
In una associazione binaria in pratica la cardinalità indica quante volte un'istanza dell'entità può essere legata a istanze dell'altra entità.
Le cardinalità minima e massima delle partecipazioni di entità a associazioni si specificano tra parentesi separate da una virgola
La cardinalità molti viene indicata con N.

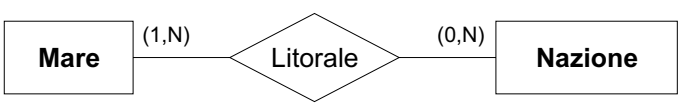
Esempio 4.8

Associazione *Residenza* tra le entità *Persona* e *Città*.
Ogni persona può partecipare a una sola associazione cioè deve risiedere in una città; ogni città può partecipare a molte associazioni cioè in una città possono risiedere più persone (o anche nessuna).



Esempio 4.9

Associazione *Litorale* tra le entità *Mare* e *Nazione*.
Ogni mare deve partecipare



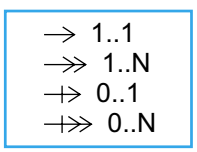
Loescher Editore - Vietata la vendita e la diffusione

ad almeno una associazione cioè deve bagnare una nazione e può partecipare a molte associazioni cioè può bagnare più nazioni; ogni nazione può partecipare a molte associazioni cioè una nazione può essere bagnata da più mari (o anche nessuno).

Molteplicità in UML

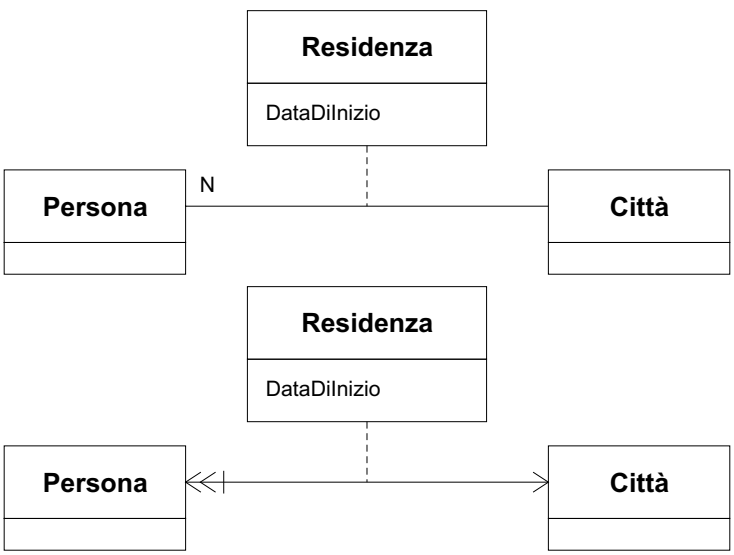
In UML le molteplicità, specificate per ciascuna entità, indicano il numero minimo e massimo di istanze dell'entità che possono partecipare a una associazione; in una associazione binaria in pratica la molteplicità indica il numero di istanze di un'entità che possono essere collegate a una istanza dell'altra entità.
Dalla parte molti sulla linea di associazione si possono aggiungere delle linee a ventaglio (zampa di corvo o crow's foot). Le molteplicità minima e massima delle entità si specificano con una coppia di valori separata da punti (..).
La molteplicità 1..1 è il valore di default e può essere omessa, o indicata con il solo valore 1.
La molteplicità molti viene indicata con N o con un asterisco; quando si specifica solo N o * si intende 0..N o 0..*.

È possibile indicare la molteplicità anche mediante punte di freccia poste al termine della linea di associazione:



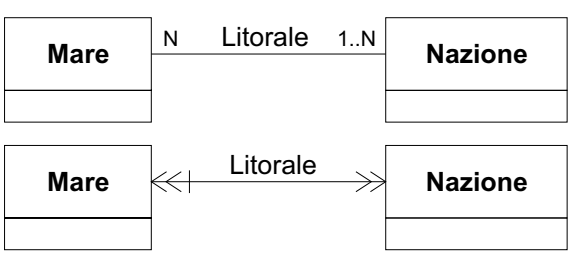
Esempio 4.10

Associazione *Residenza* tra le entità *Persona* e *Città*. Ogni persona può partecipare a una sola associazione cioè deve risiedere in una città; ogni città può partecipare a molte associazioni cioè in una città possono risiedere più persone (o anche nessuna).



Esempio 4.11

Associazione *Litorale* tra le entità *Mare* e *Nazione*.
Ogni mare deve partecipare ad almeno una associazione cioè deve bagnare una nazione e può partecipare a molte associazioni cioè può bagnare più nazioni; ogni nazione può partecipare a molte associazioni cioè una nazione può essere bagnata da più mari (o anche nessuno).



Loescher Editore - Vietata la vendita e la diffusione

Differenze tra modello ER e UML

La differenza più importante tra modello ER e rappresentazione UML è che nello schema ER le cardinalità di partecipazione di una entità vengono riportate accanto all'entità stessa, mentre nel diagramma UML le molteplicità vengono riportate accanto all'entità opposta, cioè tra uno schema ER e un diagramma UML le cardinalità/molteplicità delle associazioni binarie nei diagrammi risultano invertite.

Significato della cardinalità/molteplicità

La **cardinalità/molteplicità minima** di solito è 0 o 1: se il valore minimo è **0** la partecipazione è **facoltativa** (o **opzionale**); se il valore minimo è **1** la partecipazione è **obbligatoria**.

La **cardinalità/molteplicità massima** definisce la **cardinalità della partecipazione** all'associazione; il valore può essere 1 o molti.

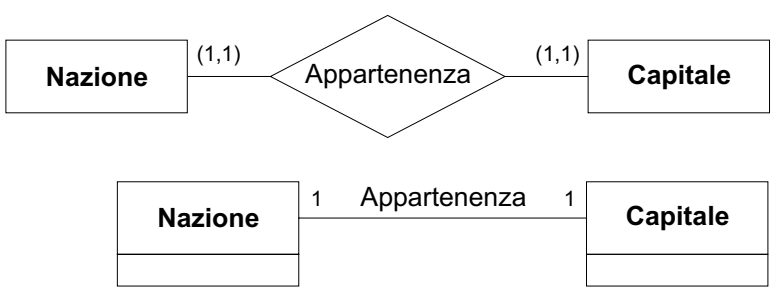
In base al tipo di corrispondenza definita dalle cardinalità/molteplicità massime le associazioni binarie possono essere classificate come:

- **1:1 uno a uno** (**biunivoca**): associa a una istanza dell'entità una sola istanza dell'altra entità, o nessuna se la partecipazione è opzionale.
- **1:N uno a molti** (**semplice**): associa a una istanza della prima entità un numero arbitrario di istanze dell'altra entità;
- **N:N molti a molti** (**complessa**): associa a una istanza della prima entità un numero arbitrario di istanze dell'altra entità e a ogni istanza della seconda entità un numero arbitrario di istanze della prima.

Nelle relazioni n-arie le entità partecipano quasi sempre con cardinalità/molteplicità massima N.

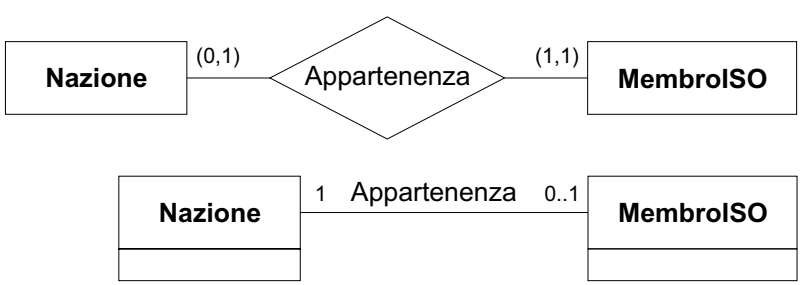
Esempio 4.12

Associazione 1:1 *Appartenenza* tra *Nazione* e *Capitale*. Ogni nazione ha una capitale; ogni capitale appartiene a una nazione. La partecipazione è obbligatoria per entrambe.



Esempio 4.13

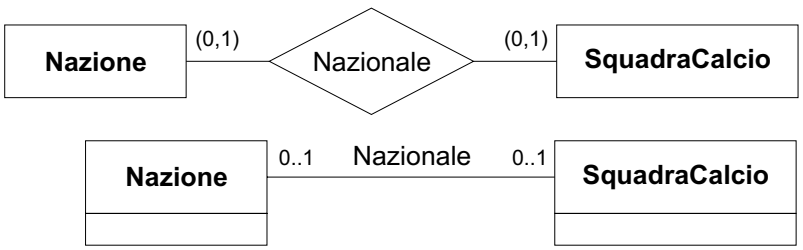
Associazione 1:1 *Appartenenza* tra *Nazione* e *MembroISO*.
Ogni nazione può essere membro dell'ISO; ogni membro dell'ISO appartiene a una nazione.
La partecipazione è facoltativa per *Nazione* (non tutte le nazioni hanno un membro all'ISO) e obbligatoria per *MembroISO*.



Loescher Editore - Vietata la vendita e la diffusione

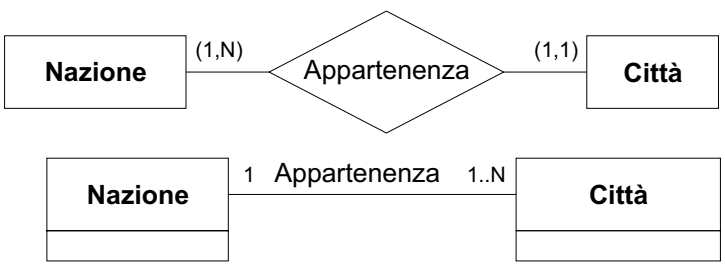
Esempio 4.14

Associazione 1:1 *Nazionale* tra *Nazione* e *SquadraCalcio*.
Ogni nazione può avere una squadra di calcio nazionale; ogni squadra di calcio appartiene a una nazione ma solo alcune sono squadre nazionali.
La partecipazione è facoltativa per entrambe.



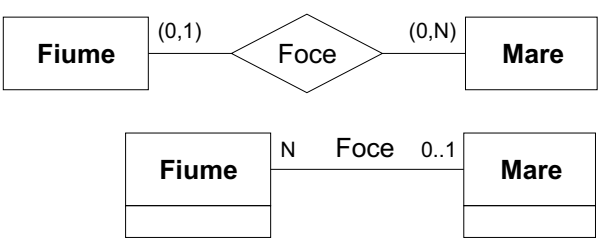
Esempio 4.15

Associazione 1:N *Appartenenza* tra *Nazione* e *Città*.
A ogni nazione appartengono una o più città, ogni città appartiene a una nazione.
La partecipazione è obbligatoria per entrambe.
Le associazioni *Appartenenza* tra *Città* e *Nazione* o tra *Nazione* e *Città* sono equivalenti; in entrambi i casi si parla di associazioni uno a molti (non si usa la dicitura molti a uno).



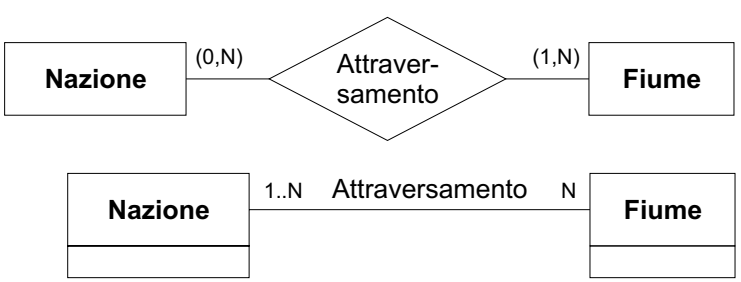
Esempio 4.16

Associazione 1:N *Foce* tra *Fiume* e *Mare*.
Ogni fiume può sfociare in un mare (non tutti, per esempio non quelli che sono affluenti di altri fiumi), ogni mare può avere più fiumi che vi sfociano (ma potrebbe anche non averne nessuno).
La partecipazione è facoltativa per entrambe.



Esempio 4.17

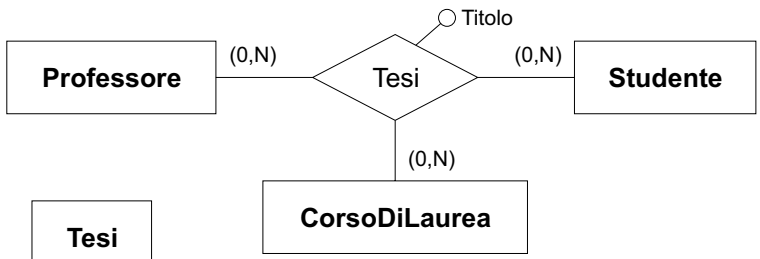
Associazione N:N *Attraversamento* tra *Nazione* e *Fiume*.
In ogni nazione possono esserci zero o più fiumi, ogni fiume può attraversare una o più nazioni.
La partecipazione è facoltativa per la nazione, obbligatoria per i fiumi.

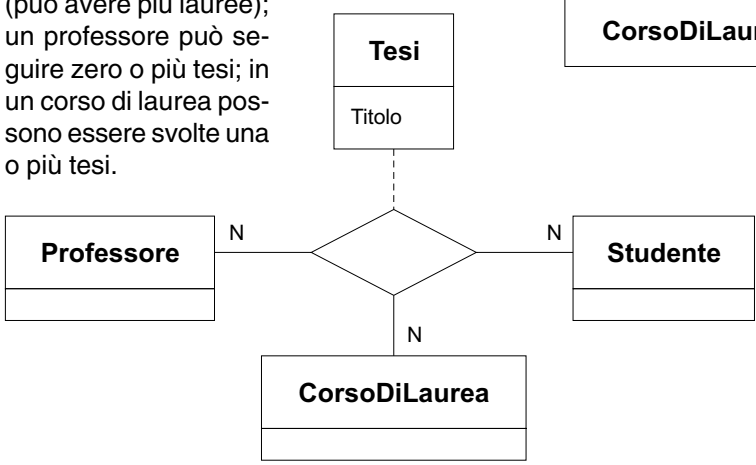


Loescher Editore - Vietata la vendita e la diffusione

Esempio 4.18

Associazione ternaria *Tesi* tra le entità *Studente*, *Professore* e *CorsoDiLaurea*.
Uno studente può svolgere zero o più tesi (può avere più lauree); un professore può seguire zero o più tesi; in un corso di laurea possono essere svolte una o più tesi.





Associazioni totali o parziali, univoche o multiple

Se la partecipazione di una entità E a una associazione è obbligatoria si dice anche che l'associazione è **totale**: il legame tra le entità deve essere sempre presente, cioè in una associazione binaria ad ogni elemento dell'entità E deve corrispondere almeno un elemento dell'altra entità.
Se la partecipazione di una entità E a una associazione è opzionale si dice anche che l'associazione è **parziale**: in una associazione binaria può esistere un elemento dell'entità E a cui non corrisponde nessun elemento dell'altra entità.

Nota Un'associazione può essere totale in un verso e parziale nell'altro: la diretta può essere totale e l'inversa parziale o viceversa.

Un'associazione parziale può essere rappresentata con una linea tratteggiata.

Una associazione si dice **univoca** se a ogni istanza di un'entità associa una sola istanza di un'altra entità, altrimenti si dice **multipla.**
Nelle associazioni uno a uno sia la diretta che l'inversa sono univoche.
Nelle associazioni uno a molti è multipla la diretta o l'inversa, ma non entrambe.
Nelle associazioni molti a molti è multipla sia la diretta che l'inversa.

Regole di lettura

Si possono descrivere le associazioni usando frasi del linguaggio naturale o **regole di lettura**.

Nota Le regole di lettura permettono di determinare la bontà della modellazione.

Loescher Editore - Vietata la vendita e la diffusione

In **ER** in base alla **cardinalità dell'entità di partenza** e in **UML** in base alla **molteplicità dell'entità di destinazione** si usano le parole:
1: un / un solo
N : molti / uno o più

uno a uno: un x verboAttivo un y
un y verboPassivo un x

verboAttivo e *verboPassivo* sono i nomi dei due versi dell'associazione, se per i nomi si usando verbi; se per i nomi delle associazioni si usano sostantivi sono comunque due verbi, attivo e passivo, ricavati dal nome dell'associazione.

uno a molti: un x verboAttivo molti / uno o più y
un y verboPassivo un / un solo y

molti a molti: un x verboAttivo molti / uno o più y
un y verboPassivo molti / uno o più x

Una nazione ha una capitale.
Ogni capitale appartiene a una nazione.

Una città appartiene a una sola nazione.
Una nazione ha molte città.

Un fiume attraversa una o più nazioni.
Una nazione è attraversata da molti fiumi.

In base all'**obbligatorietà** o meno della partecipazione si possono aggiungere le parole può o deve:

partecipazione **obbligatoria**: x deve nomeAssociazione un solo / uno o più y

partecipazione **opzionale**: x può nomeAssociazione un solo / uno o più y

per *nomeAssociazione* si intende il verbo attivo o passivo che rappresenta il verso dell'associazione.

Una nazione deve avere una capitale.
Una capitale deve appartenere a una nazione.

Una nazione può essere un membro dell'ISO.
Un membro dell'ISO deve essere l'organismo di una nazione.

Una nazione può avere una nazionale di calcio.
Una squadra di calcio può essere la nazionale per una nazione.

A una nazione devono appartenere una o più città.
Una città deve appartenere a una sola nazione.

Un fiume può sfociare in un mare.
In un mare possono sfociare molti fiumi.

Un fiume deve attraversare una o più nazioni.
Una nazione può essere attraversata da molti fiumi.

Loescher Editore - Vietata la vendita e la diffusione

RIASSUMENDO

nome dell'entità di partenza
deve / può
nome dell'associazione (verbo attivo o passivo)
un solo / uno o più
nome dell'entità di destinazione

deve / può: dipende dall'obbligatorietà o meno della partecipazione dell'entità;
un solo / uno o più: dipende in ER dalla cardinalità di partenza, in UML dalla molteplicità di destinazione.

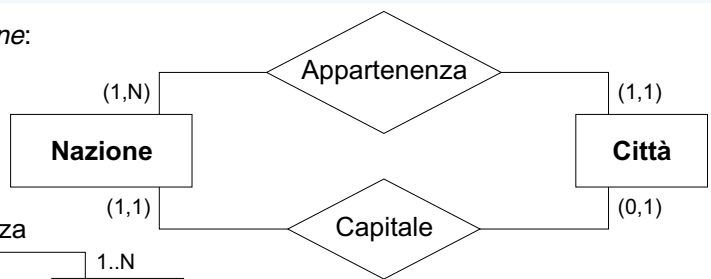
un x deve / può nomeAssociazione un solo / uno o più y

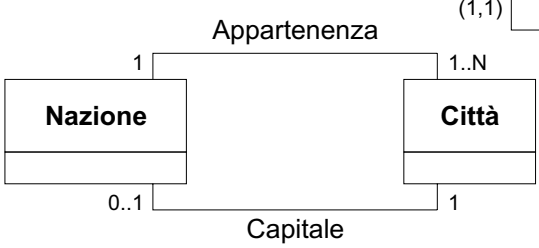
RUOLI DELLE ENTITÀ NELLE ASSOCIAZIONI

Tra le stesse entità possono esserci più associazioni diverse; si possono distinguere mediante un nome i **ruoli** che le entità hanno in ciascuna associazione.

ESEMPIO 4.19

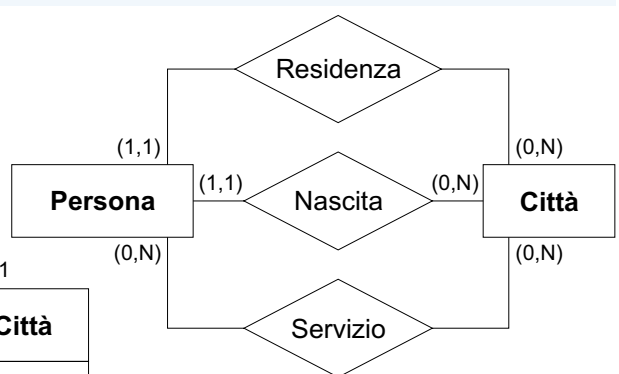
Associazioni tra *Città* e *Nazione*:
- *Appartenenza*
- *Capitale*

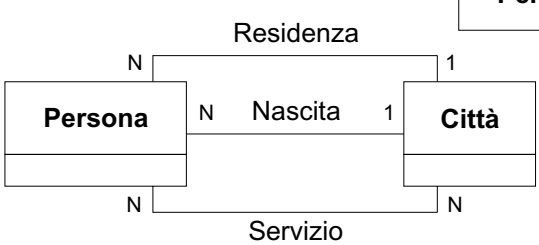




ESEMPIO 4.20

Associazioni tra *Persona* e *Città*:
- *Residenza*
- *Nascita*
- *Servizio*





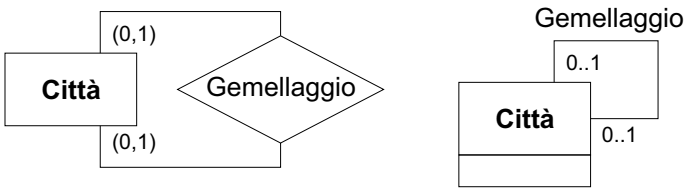
Loescher Editore - Vietata la vendita e la diffusione

Associazioni ricorsive

Si possono avere associazioni **ricorsive** tra una entità e se stessa.
L'entità può partecipare all'associazione con lo stesso ruolo (associazione ricorsiva **simmetrica**) o con ruoli diversi (associazione ricorsiva **non simmetrica**); se l'associazione non è simmetrica è importante distinguere i ruoli associando degli identificatori alle linee uscenti dalla associazione ricorsiva.
Anche le associazioni ricorsive possono essere uno a uno, uno a molti o molti a molti.

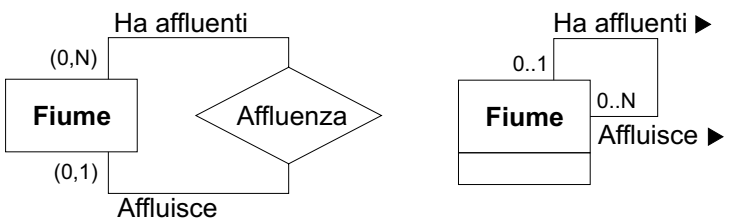
Esempio 4.21

Associazione *Gemellaggio* tra le città (simmetrica uno a uno).



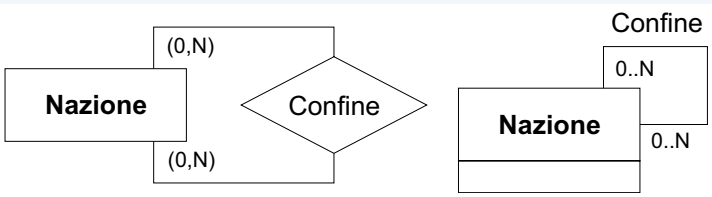
Esempio 4.22

Associazione *Affluenza* tra i fiumi (non simmetrica uno a molti).



Esempio 4.23

Associazione *Confine* tra le nazioni (simmetrica molti a molti).



Generalizzazioni

Una **generalizzazione** è un legame logico tra più entità, di cui una è detta padre e le altre figlie; l'entità padre è più generale, nel senso che comprende le entità figlie come casi particolari.

La generalizzazione è anche chiamata associazione **ISA** (is a).
Possono esserci più livelli di associazioni ISA e si parla in questo caso di gerarchie ISA.

L'entità padre è la generalizzazione delle entità figlie.
Le entità **figlie** sono **particolarizzazioni** o **specializzazioni** dell'entità padre.

Le entità figlie sono sottoinsiemi dell'entità padre: ogni istanza di una entità figlia è anche una istanza dell'entità padre.

Tra le entità figlie e l'entità padre vale una relazione di **ereditarietà**: le entità figlie ereditano tutte le proprietà dell'entità padre (attributi, identificatori, associazioni e altre generalizzazioni): ogni proprietà dell'entità padre è anche una proprietà delle entità figlie.

Nelle figlie le proprietà ereditate non vanno rappresentate esplicitamente.

Loescher Editore - Vietata la vendita e la diffusione

Nel **modello ER** le generalizzazioni sono rappresentate da frecce che congiungono le entità figlie con l'entità padre.
In **UML** le generalizzazioni sono rappresentate con la freccia di ereditarietà: una freccia con la punta a triangolo vuoto che va dall'entità figlia all'entità padre.

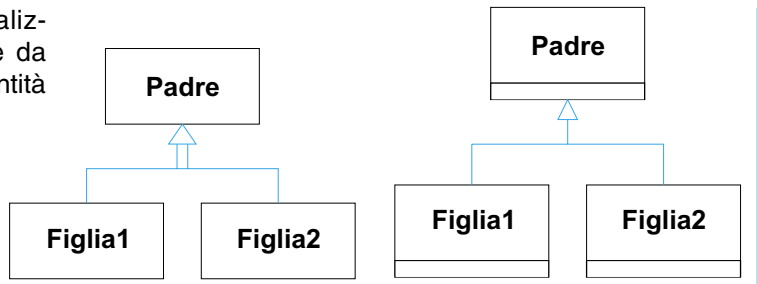


Figura 4.7 *Rappresentazione della generalizzazione*

Le gerarchie ISA si rappresentano con un albero.

Una stessa entità può essere coinvolta in più generalizzazioni diverse.

Le generalizzazioni possono essere classificate in base a due proprietà:

- una generalizzazione è **totale** (complete) se ogni istanza dell'entità padre è una istanza di almeno una delle figlie, è **parziale** (incomplete) se ci possono essere istanze dell'entità padre che non appartengono a nessuna delle figlie.

Nota

A volte una generalizzazione è parziale perché si definiscono solo le entità necessarie all'applicazione.

Nel **modello ER** le generalizzazioni totali sono rappresentate con una freccia con il tratto pieno, le generalizzazioni parziali con una freccia con il tratto vuoto.
In **UML** la freccia dell'ereditarietà può riportare le **proprietà di partizionamento** (tra parentesi graffe).

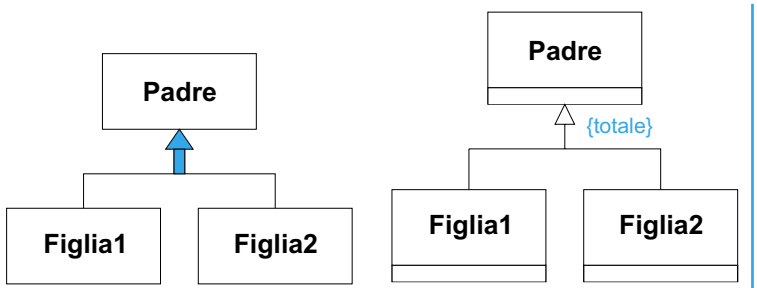
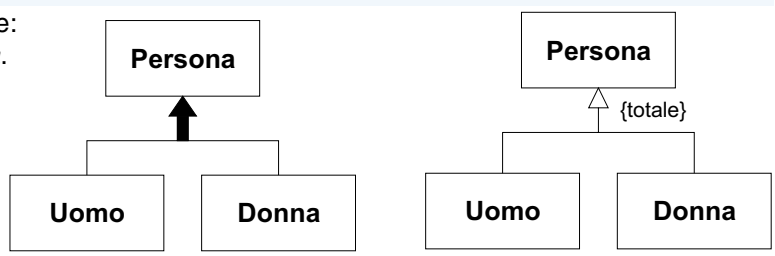


Figura 4.8 *Rappresentazione della generalizzazione totale*

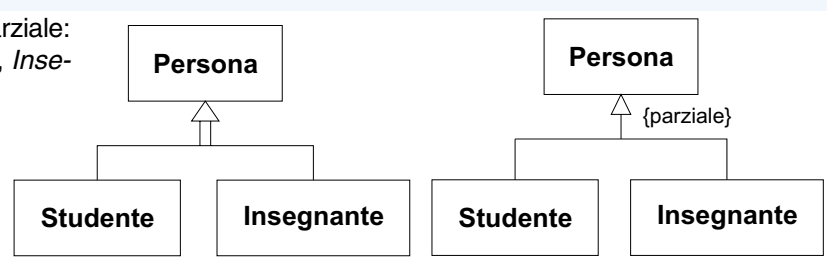
Esempio 4.24

Generalizzazione totale: *Persona - Uomo, Donna.*



Esempio 4.25

Generalizzazione parziale: *Persona - Studente, Insegnante.*

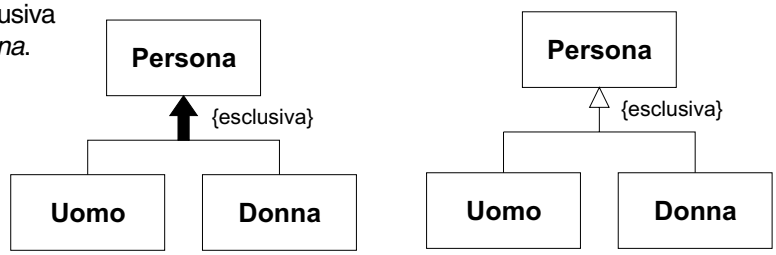


Loescher Editore - Vietata la vendita e la diffusione

- una generalizzazione è **esclusiva** o disgiunta (disjoint) se ogni istanza dell'entità padre è al più una istanza di una delle entità figlie, è **sovrapposta** (overlapping) se qualche istanza dell'entità padre può appartenere contemporaneamente a più di una entità figlia.

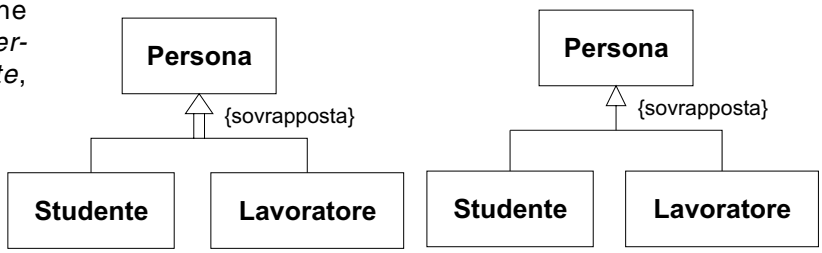
Esempio 4.26

Generalizzazione esclusiva *Persona - Uomo, Donna*.



Esempio 4.27

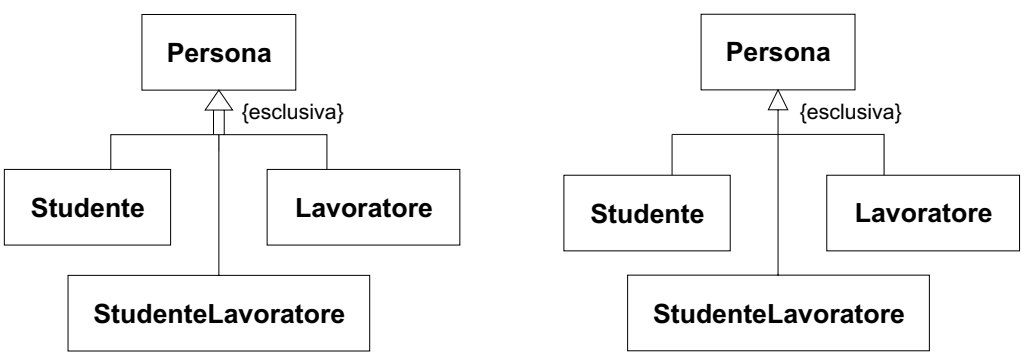
Generalizzazione sovrapposta: *Persona - Studente, Lavoratore*.



Le generalizzazioni sovrapposte possono essere trasformate in esclusive aggiungendo entità figlie che rappresentano le intersezioni, quindi si può fare l'ipotesi che le generalizzazioni siano sempre esclusive.

Esempio 4.28

Generalizzazione sovrapposta trasformata in esclusiva: *Persona - Studente, Lavoratore, StudenteLavoratore*.



Identificatore esterno

L'identificatore di una entità è esterno quando l'entità è identificata anche mediante attributi di un'altra entità.

Loescher Editore - Vietata la vendita e la diffusione

Nel modello ER l'identificatore esterno viene rappresentato con un cerchio nero collegato da una linea alla linea che rappresenta l'associazione.

In UML l'identificatore esterno viene rappresentato con lo stereotipo <<identificante>> sulla linea dell'associazione.

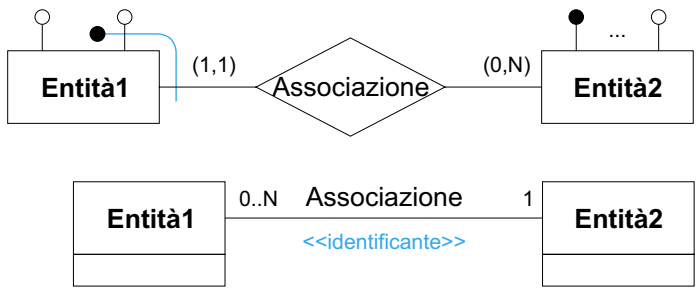
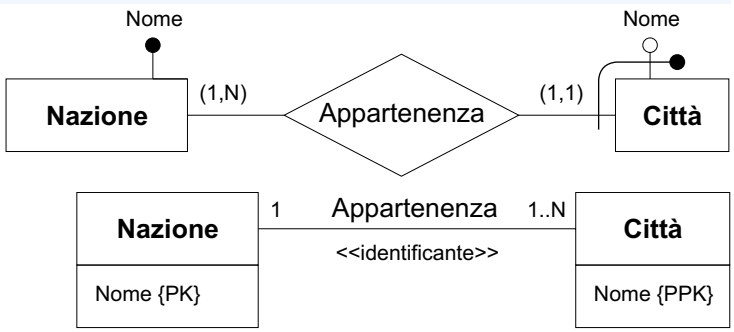


Figura 4.9 *Rappresentazione di un identificatore esterno*

Esempio 4.29

La *Città* è identificata oltre che dal *Nome* anche dalla *Nazione* a cui appartiene (altrimenti ci potrebbero essere nomi duplicati).

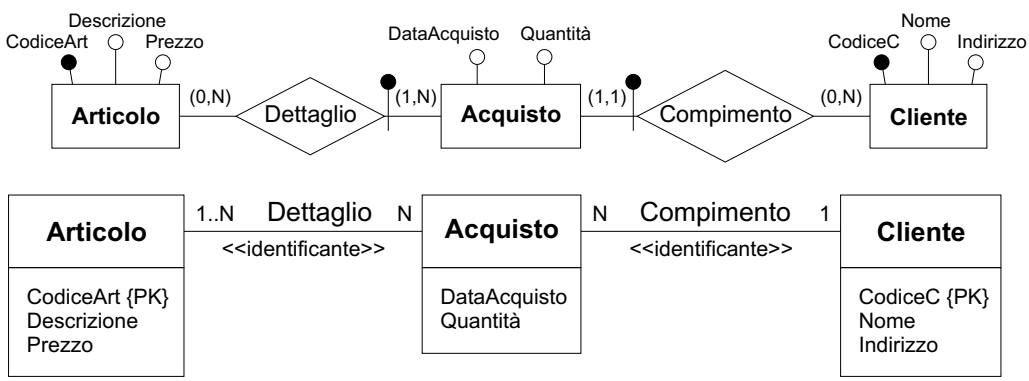


Entità forti e entità deboli

Se una entità ha una chiave primaria è detta **entità forte**.
Una entità priva di chiave primaria, che necessita di altre entità a cui è legata da associazioni per essere significativa, è detta **entità debole**.

Esempio 4.30

Entità debole *Acquisto* identificata da *Cliente* e *Articolo* acquistato.



Per evitare entità deboli spesso si aggiunge un attributo con un numero progressivo assegnato automaticamente.

Si può aggiungere all'entità *Acquisto* dell'esempio 4.30 l'attributo *IdAcquisto* con un numero progressivo.

Loescher Editore - Vietata la vendita e la diffusione

4.5 REGOLE AZIENDALI

Le proprietà di una applicazione che non possono essere rappresentare direttamente nel modello concettuale, possono essere descritte mediante **regole aziendali** (business rule).

Il termine **regola aziendale** viene usato in modo generico per indicare qualunque informazione che definisce o vincola qualche aspetto dell'applicazione.

Esempi di regole aziendali sono:

- la **descrizione precisa** di un'entità, un attributo o un'associazione;

  per le descrizioni si fa ricorso a frasi del linguaggio naturale;
- una **derivazione**, cioè un concetto che può essere ottenuto attraverso un'inferenza o un calcolo aritmetico da altri concetti dello schema;

  le regole di derivazione possono essere espresse come:
  <concetto> si ottiene <operazione su concetti>

  > La densità abitativa di una nazione si ottiene dividendo la popolazione per la superficie.
  >
  > La popolazione di un continente si ottiene sommando la popolazione delle nazioni che appartengono al continente.
- un **vincolo di integrità**.

VINCOLI DI INTEGRITÀ

Un **vincolo di integrità** è una proprietà che deve essere soddisfatta dalle istanze di entità e associazioni per essere valide.

Nota I vincoli di integrità devono essere rispettati pena l'integrità dei dati.

Ci sono **vincoli impliciti** imposti dalla struttura dei dati e **vincoli espliciti** imposti con apposite dichiarazioni.

VINCOLI IMPLICITI

Esempi di vincoli impliciti sono:

- **cardinalità** degli attributi e delle associazioni;
- **vincoli di chiave primaria**: impongono che le istanze di una entità siano tutte diverse tra loro;
- **vincoli referenziali**: date due entità e una associazione tra queste, impongono che non si possa inserire un elemento nella prima entità che non sia associato a un elemento della seconda (vincolo sulla totalità delle associazioni).

I vincoli impliciti si rappresentano direttamente nel diagramma:

- la cardinalità con i numeri relativi;
- i vincoli di chiave primaria evidenziando la chiave (con sottolineatura, cerchio nero o {PK});
- i vincoli referenziali con le linee continue dell'associazione totale.

VINCOLI ESPLICITI

I vincoli espliciti vengono espressi sotto forma di **asserzioni** o **predicati**, cioè affermazioni che devono sempre essere verificate.

Loescher Editore - Vietata la vendita e la diffusione

Vanno enunciati in linguaggio naturale o con uno pseudolinguaggio in una forma dichiarativa, cioè che non suggerisca un metodo per soddisfarli.

Nota

Forme del tipo “se condizione allora azione” non vanno bene, perché espresse in modo procedurale e non dichiarativo.

<concetto> deve/non può <espressione su concetti>
dove i concetti sono elementi che compaiono nella schema o ne sono derivabili.
Si può fare riferimento agli attributi con la notazione puntata *NomeEntità.NomeAttributo* o *NomeAssociazione.NomeAttributo*.

La superficie di una nazione deve essere un numero positivo.

La popolazione di una città non può essere superiore alla popolazione della nazione a cui la città appartiene.

Nota

L’esistenza dei vincoli richiede che il DBMS abbia meccanismi in grado di controllare che i vincoli siano rispettati, impedendo tentativi di violazione.

4.6 PRINCIPI DI MODELLAZIONE

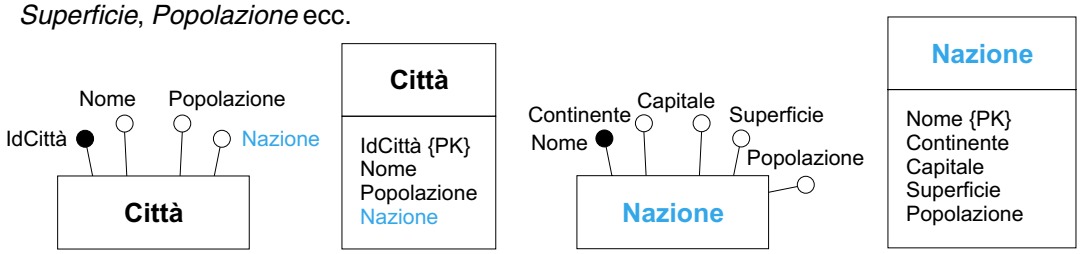
La scelta di modellare un concetto come entità, attributo o associazione dipende dal contesto della realtà di interesse e dall’uso che se ne vuole fare.

Entità e attributi

Un concetto deve essere rappresentato come attributo se è un unico valore, come entità se sono necessarie anche altre informazioni, descritte nelle proprietà che la caratterizzano.

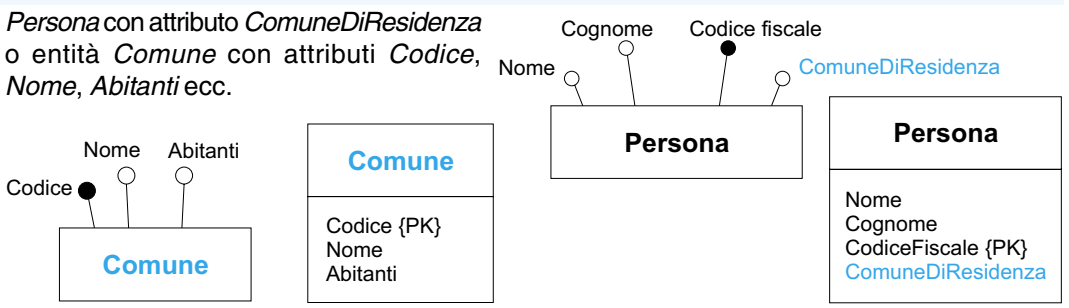
Esempio 4.31

Città con attributo *Nazione* o entità *Nazione* con attributi *Nome*, *Continente*, *Capitale*, *Superficie*, *Popolazione* ecc.



Esempio 4.32

Persona con attributo *ComuneDiResidenza* o entità *Comune* con attributi *Codice*, *Nome*, *Abitanti* ecc.

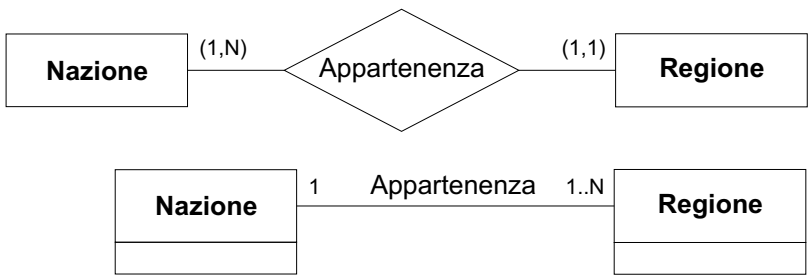


Loescher Editore - Vietata la vendita e la diffusione

Un attributo multivalore deve essere modellato come entità legata da una associazione uno a molti o molti a molti con l'entità cui si riferisce.

Esempio 4.33

Nazione con attributo multivalore *Regione*; *Regione* deve essere modellata come entità legata a *Nazione* dall'associazione *Appartenenza*; in questo modo inoltre è possibile gestire altre informazioni sulle regioni (come capoluogo, superficie ecc.)

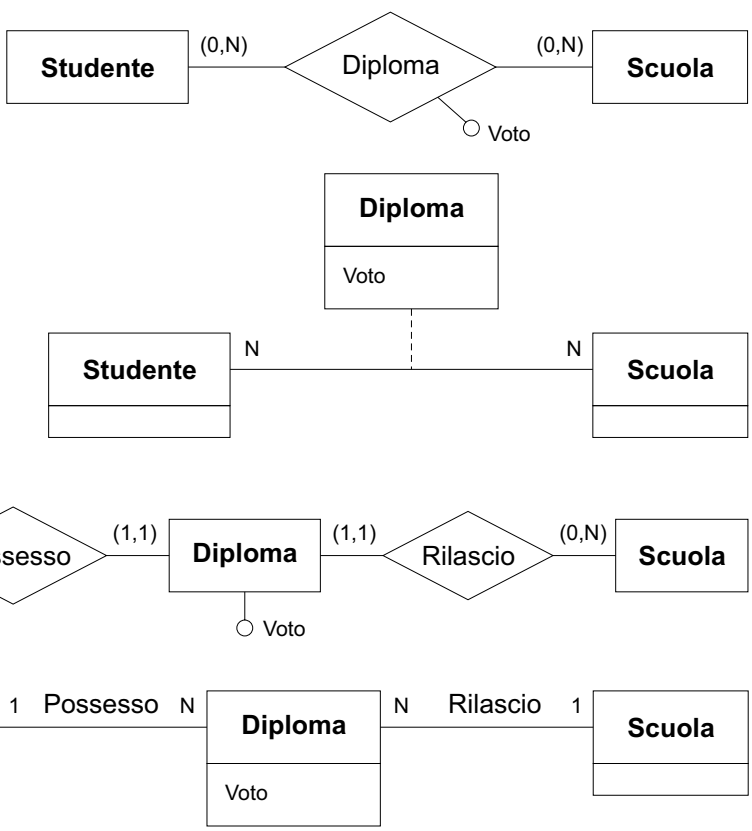


Associazioni con attributi e entità

Si possono eliminare gli attributi di una associazione introducendo una nuova entità.

Esempio 4.34

Associazione *Diploma* tra *Studente* e *Scuola*; l'associazione *Diploma* ha come attributo il voto; per eliminarlo si può introdurre una nuova entità *Diploma*.

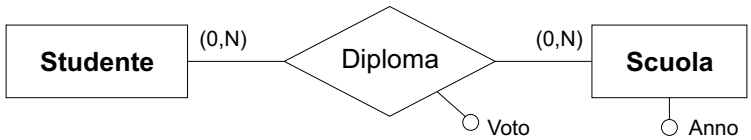


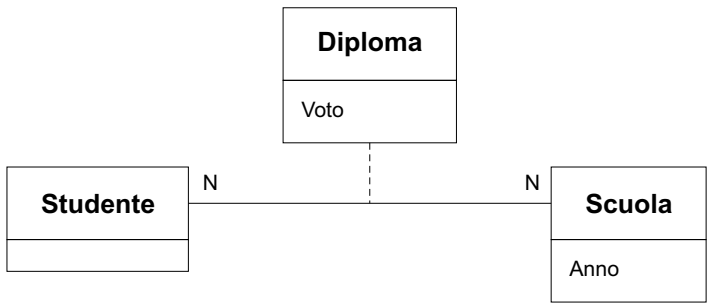
Loescher Editore - Vietata la vendita e la diffusione

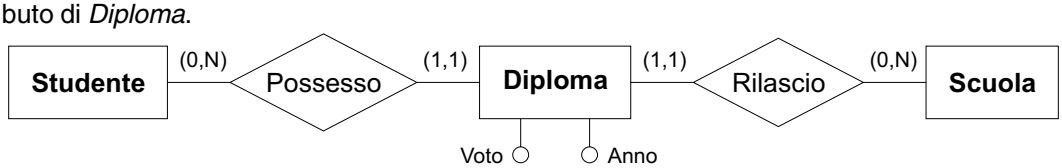
Se un attributo ha un valore fisso non è mai attributo di una associazione ma di una entità.

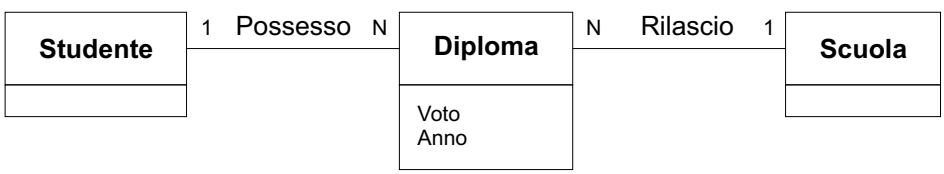
Esempio 4.35

Associazione *Diploma* tra *Studente* e *Scuola*; se si considerano i diplomi di un anno, *Anno* non è attributo dell'associazione *Diploma* ma dell'entità *Scuola*. È possibile eliminare l'attributo dell'associazione creando una nuova entità *Diploma*; in questo caso si può decidere di far diventare anche *Anno* attributo di *Diploma*.





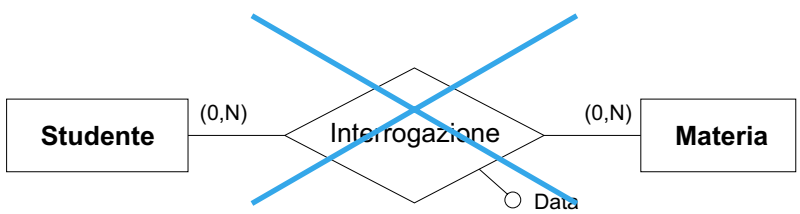


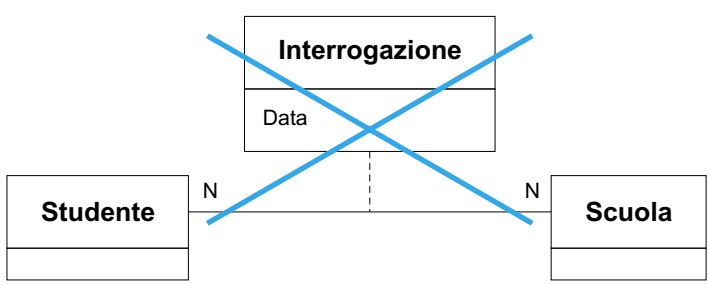


A volte la scelta tra associazioni e entità è obbligatoria.

Esempio 4.36

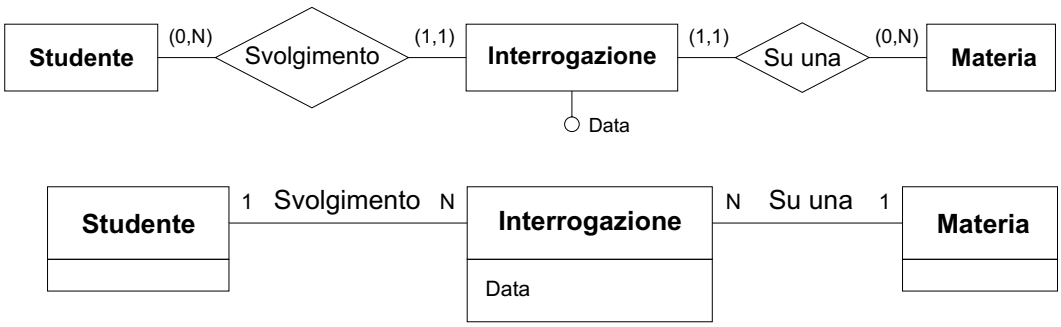
Interrogazione di uno *Studente* in una *Materia*.
Interrogazione non può essere modellata come associazione perché uno studente può essere interrogato più volte nella stessa materia; non basta considerare la data come attributo dell'associazione perché si avrebbero co-





Loescher Editore - Vietata la vendita e la diffusione

munque più istanze dell'associazione formate dalla stessa coppia (studente, materia) e questo contraddice la definizione di associazione come insieme (non possono esserci valori duplicati).
La modellazione corretta è quindi:

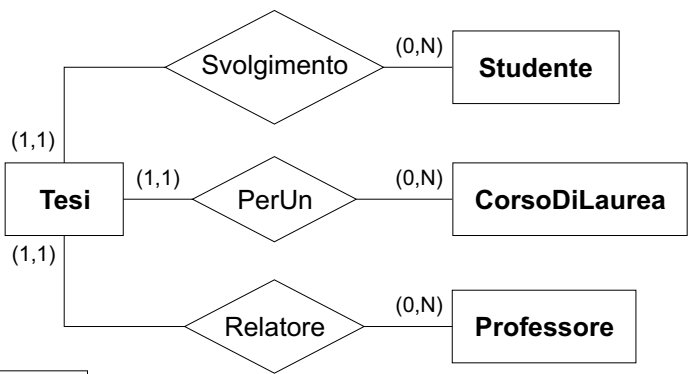


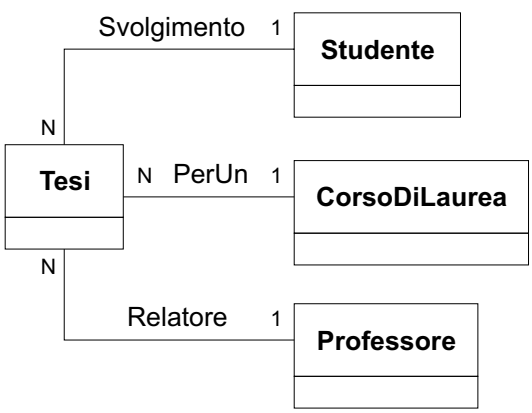
Associazioni multiple

Le associazioni multiple possono sempre essere trasformate in più relazioni binarie: l'associazione viene trasformata in entità e vengono aggiunte associazioni per rappresentare i collegamenti con le altre entità.

Esempio 4.37

L'associazione ternaria *Tesi* dell'esempio 4.7 può essere trasformata nell'entità *Tesi* considerando le associazioni binarie *Svolgimento* con *Studente*, *PerUn* con *CorsoDiLaurea* e *Relatore* con *Professore*.





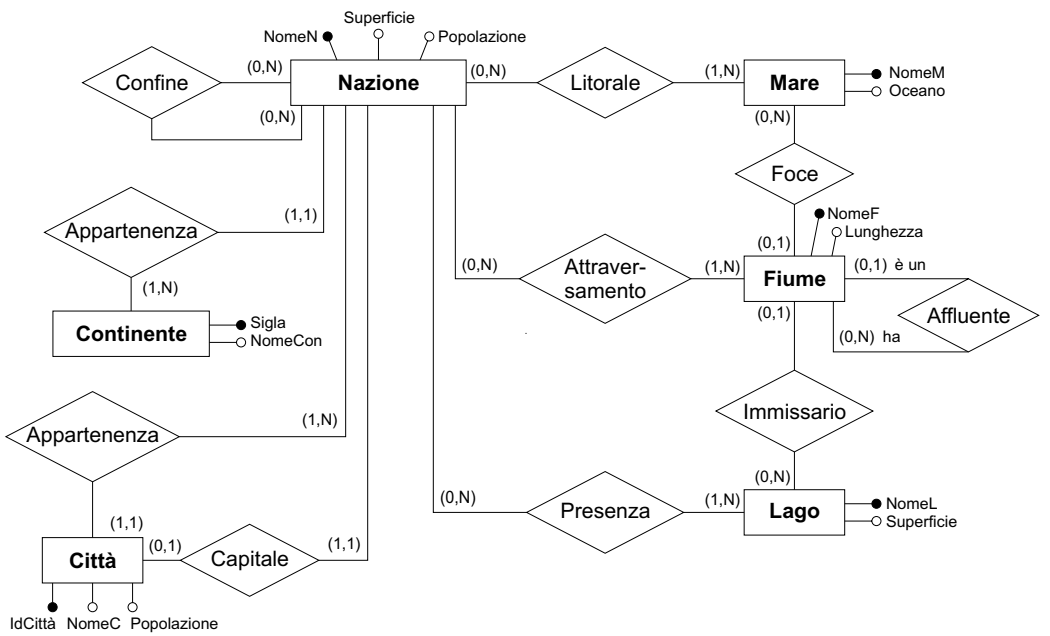
Diagrammi equivalenti

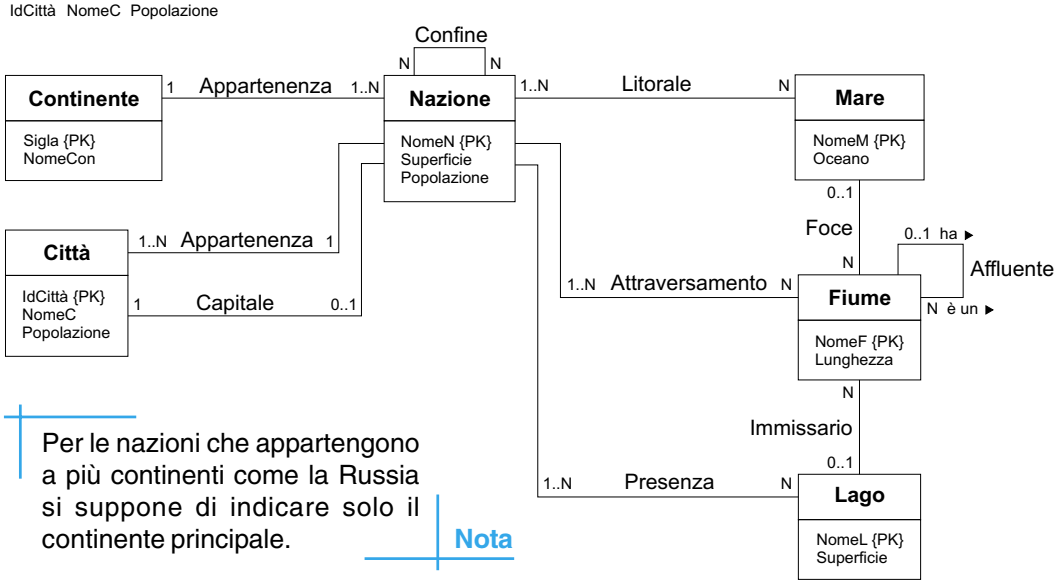
La stessa realtà può essere rappresentata con diagrammi diversi equivalenti.
Due diagrammi sono **equivalenti** se hanno lo stesso contenuto informativo, cioè se per ogni interrogazione su un diagramma esiste una interrogazione sull'altro diagramma che fornisce la stessa risposta.
Diagrammi equivalenti possono differire per caratteristiche come chiarezza, leggibilità e comprensibilità o sinteticità e compattezza.

4.7 ESEMPI SVOLTI

Esempio 4.38

Geografia: gestione dei dati relativi a nazioni, città, mari, fiumi e laghi.



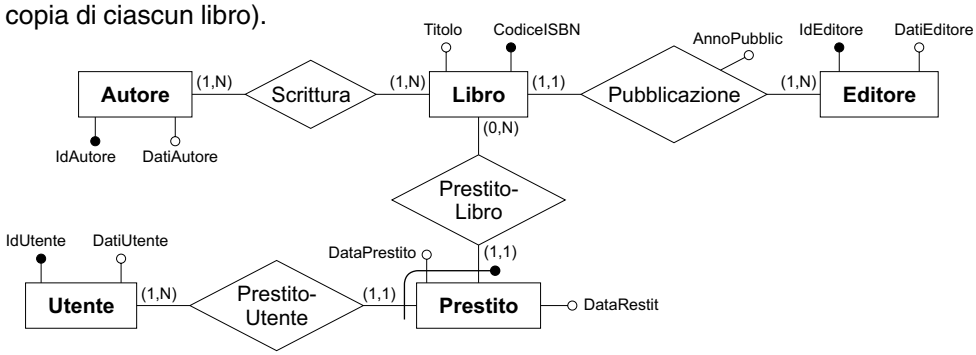


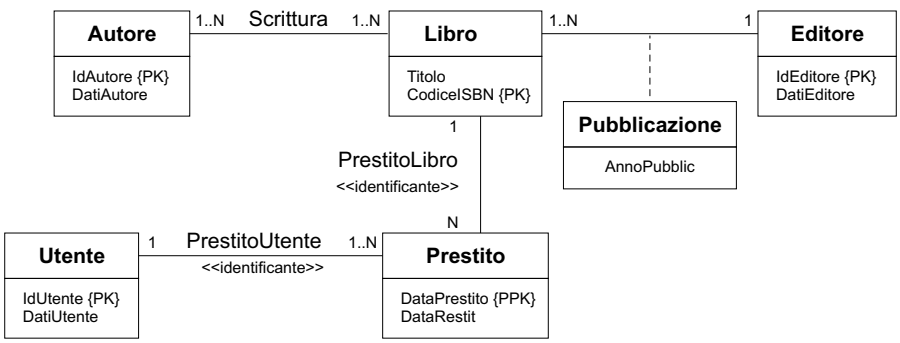
Per le nazioni che appartengono a più continenti come la Russia si suppone di indicare solo il continente principale.

Nota

Esempio 4.39

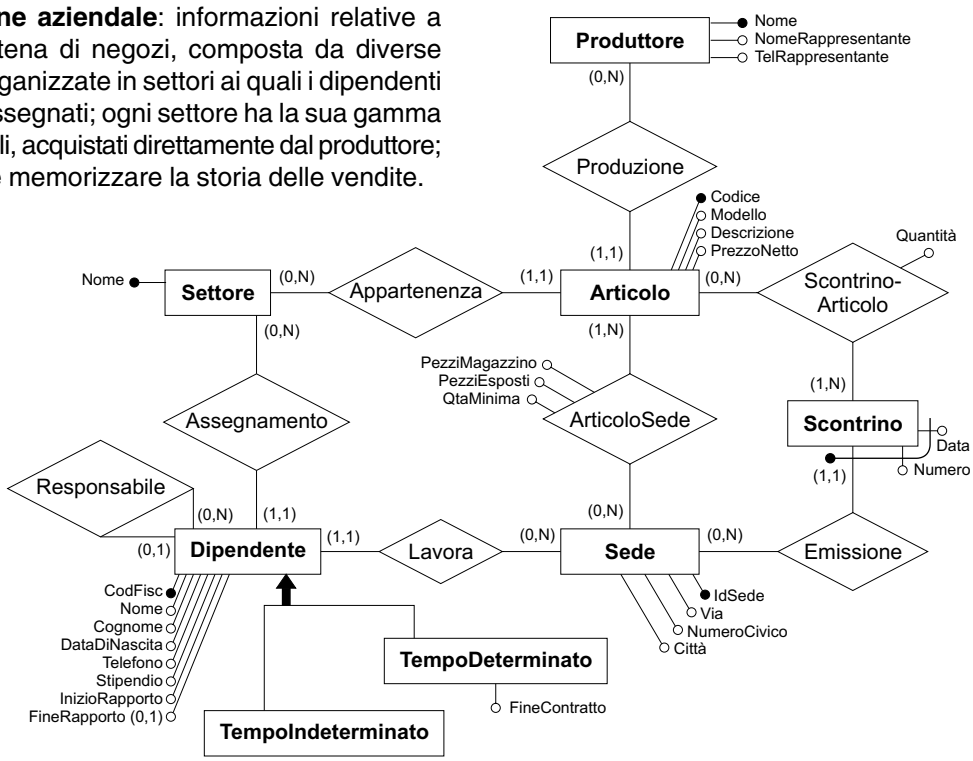
Biblioteca: gestione dei prestiti dei libri presenti in una biblioteca (è disponibile una sola copia di ciascun libro).



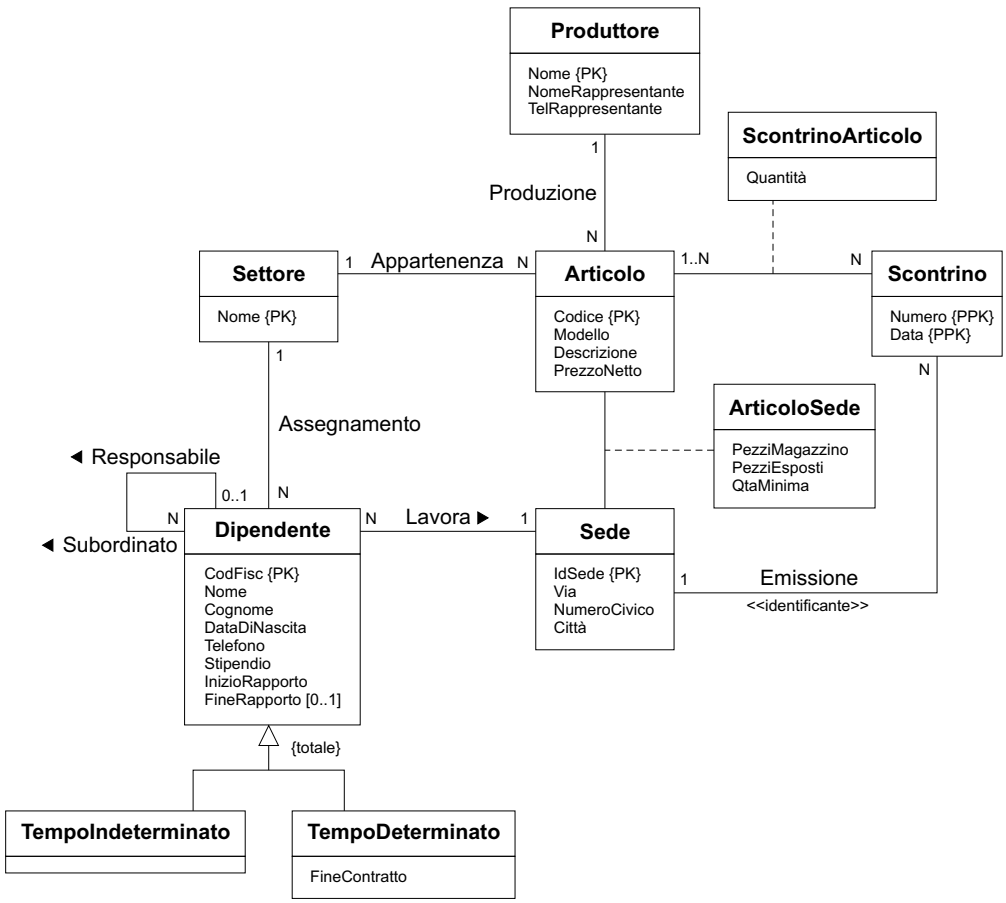


Esempio 4.40

Gestione aziendale: informazioni relative a una catena di negozi, composta da diverse sedi, organizzate in settori ai quali i dipendenti sono assegnati; ogni settore ha la sua gamma di articoli, acquistati direttamente dal produttore; si vuole memorizzare la storia delle vendite.



Loescher Editore - Vietata la vendita e la diffusione



VERIFICA

QUESTIONARIO

1. Che cos'è lo schema concettuale di un database?
2. Come si può rappresentare lo schema concettuale?
3. Quali sono i principali tipi di diagrammi UML e che cosa permettono di rappresentare?
4. Che cos'è una entità?
5. Che cosa sono gli attributi?
6. Che cos'è un identificatore di una entità?
7. Che cosa sono le associazioni?
8. Che cos'è la cardinalità o molteplicità delle associazioni?
9. Che cosa si intende per generalizzazione?
10. Che cosa sono le entità deboli?
11. Che cos'è un identificatore esterno?
12. Che cosa sono le regole aziendali?
13. Che cosa sono i vincoli di integrità?

Loescher Editore - Vietata la vendita e la diffusione

TEST

1 ..... *Indicare se le seguenti affermazioni sono vere o false.*

V	F	
☐	☐	1) L'insieme delle istanze di una entità in un certo momento è l'insieme di tutte le possibili istanze di quell'entità.
☐	☐	2) Gli attributi rappresentano le proprietà di una entità.
☐	☐	3) Gli attributi composti possono contenere altri attributi composti.
☐	☐	4) Un attributo enumerativo può assumere un valore compreso in un intervallo infinito.
☐	☐	5) Un attributo è derivato se si può ottenere dal calcolo di una espressione.
☐	☐	6) L'identificatore di una entità può essere solo un attributo dell'entità.
☐	☐	7) L'identificatore ha sempre un valore diverso per ogni entità.
☐	☐	8) L'identificatore può essere costituito da un numero progressivo.
☐	☐	9) Le associazioni possono avere attributi.
☐	☐	10) Le associazioni sono sempre binarie.
☐	☐	11) In una associazione ci possono essere più ennuple uguali.
☐	☐	12) Nei diagrammi ER e UML la cardinalità delle associazioni si rappresenta allo stesso modo.
☐	☐	13) Un'associazione è totale se la partecipazione di entrambe le entità è obbligatoria.
☐	☐	14) Tra due entità possono esserci più associazioni.
☐	☐	15) Un'associazione ricorsiva non può essere molti a molti.
☐	☐	16) Una generalizzazione stabilisce un legame di ereditarietà tra l'entità padre e le entità figlie.
☐	☐	17) Una generalizzazione è totale se ogni istanza dell'entità padre è una istanza di almeno una delle figlie.
☐	☐	18) Una generalizzazione sovrapposta è sempre totale.
☐	☐	19) Un'entità forte ha un identificatore interno.
☐	☐	20) Un vincolo referenziale è un vincolo esplicito.
☐	☐	21) Due diagrammi equivalenti hanno lo stesso numero di entità.

2 ..... *Segnare tutte le affermazioni esatte.*

1) Relativamente ai diagrammi ER e UML:

☐ il modello ER è stato creato appositamente per la rappresentazione dello schema concettuale dei database;

☐ il modello ER permette di rappresentare tutti gli aspetti di una applicazione software;

☐ UML è stato creato appositamente per la rappresentazione dello schema concettuale dei database;

Loescher Editore - Vietata la vendita e la diffusione

- ☐ UML permette di rappresentare tutti gli aspetti di una applicazione software;
- ☐ UML è uno standard più recente rispetto ai diagrammi ER;
- ☐ UML offre notazioni standard per qualsiasi aspetto dello schema concettuale del database.

2) Se un attributo ha cardinalità minima 0 e cardinalità massima 1 è:
- ☐ opzionale;
- ☐ obbligatorio;
- ☐ multivalore.

3) Se un attributo ha cardinalità minima 1 e cardinalità massima 1 è:
- ☐ opzionale;
- ☐ obbligatorio;
- ☐ multivalore.

4) Se un attributo ha cardinalità minima 0 e cardinalità massima N è:
- ☐ opzionale;
- ☐ obbligatorio;
- ☐ multivalore.

5) Se un attributo ha cardinalità minima 1 e cardinalità massima N è:
- ☐ opzionale;
- ☐ obbligatorio;
- ☐ multivalore.

6) Se le cardinalità delle entità che partecipano a una associazione sono (0,1) e (1,1):
- ☐ l'associazione è 1:1;
- ☐ l'associazione è 1:N;
- ☐ l'associazione è N:N;
- ☐ la partecipazione è obbligatoria per entrambe le entità;
- ☐ la partecipazione è facoltativa per entrambe le entità;
- ☐ la partecipazione è facoltativa per una sola entità.

7) Se le cardinalità delle entità che partecipano a una associazione sono (0,N) e (0,N):
- ☐ l'associazione è 1:1;
- ☐ l'associazione è 1:N;
- ☐ l'associazione è N:N;
- ☐ la partecipazione è obbligatoria per entrambe le entità;
- ☐ la partecipazione è facoltativa per entrambe le entità;
- ☐ la partecipazione è facoltativa per una sola entità.

8) Se le cardinalità delle entità che partecipano a una associazione sono (1,N) e (1,1):
- ☐ l'associazione è 1:1;
- ☐ l'associazione è 1:N;
- ☐ l'associazione è N:N;
- ☐ la partecipazione è obbligatoria per entrambe le entità;
- ☐ la partecipazione è facoltativa per entrambe le entità;
- ☐ la partecipazione è facoltativa per una sola entità.

Loescher Editore - Vietata la vendita e la diffusione

Esercizi

1 ..... Descrivere gli attributi delle seguenti entità e individuare gli identificatori.

- Computer
- Articolo in vendita in un supermercato
- Figura geometrica
- Film
- Voce in una rubrica telefonica
- Appuntamento in una agenda di appuntamenti
- Software
- Animale

2 ..... Leggere gli esempi del capitolo con le regole di lettura.

3 ..... Rappresentare con un diagramma ER e/o UML le seguenti frasi, indicando entità, attributi, chiavi, associazioni, cardinalità/molteplicità minima e massima ecc.:

- Una persona può partecipare a zero o più viaggi; un viaggio può essere prenotato da zero o più persone.
- Un'aula può essere occupata da una classe; ogni classe deve stare in un'aula.
- A ogni classe appartengono uno o più studenti; ogni studente deve appartenere a una classe; il numero di studenti di una classe si ottiene contando gli studenti che vi appartengono.
- Ogni auto è posseduta da una persona in un determinato periodo; ogni persona può possedere zero o più auto.
- In ogni classe insegnano uno o più insegnanti, ogni insegnante insegna in una o più classi.
- Ogni classe ha più insegnanti e un insegnante coordinatore.
- Ogni studente universitario è iscritto a una università; ogni università può avere molti iscritti. (Suggerimento: il numero di matricola dello studente è identificato esternamente dall'università che frequenta).
- Ogni scheda di rete è commercializzata da un produttore. (Suggerimento: il progressivo dell'indirizzo MAC della scheda è identificato esternamente dal produttore della scheda).
- Una persona può avere un coniuge.
- Una persona è figlia di un padre e una madre e può avere zero o più figli (padre, madre e figlio sono a loro volta persone).
- Un supplente insegna in una classe da una data iniziale a una data finale.
- Un concorrente può partecipare a una gara e ottiene un risultato; a una gara possono partecipare più concorrenti.
- In una certa ora una classe si trova in un'aula e svolge una materia con un insegnante.

4 ..... Modellare correttamente l'entità:

Persona

Nome	stringa
Cognome	stringa
CodiceFiscale	stringa
Sesso	enumerativo (M, F)
DataDiNascita	elementare o composto da giorno mese e anno
Indirizzo	composto da via, numero, città, provincia e CAP
Telefono	multivalore
Hobby	multivalore opzionale (può avere valore *null*)
Età	derivato *annoCorrente - AnnoDiNascita*

Loescher Editore - Vietata la vendita e la diffusione

5 ..... Per l'esempio geografia:
Modificare il diagramma in modo da introdurre per le città informazioni sui fiumi che le attraversano, e sui mari o laghi che le bagnano.

6 ..... Per l'esempio biblioteca:
Dato lo schema attuale è possibile per un utente prendere più volte in prestito lo stesso libro?
Modificare il diagramma in modo da introdurre:
- la possibilità di avere più copie dello stesso libro;
- la classificazione per argomento (un libro può riguardare più argomenti);
- informazioni sulla disponibilità e la possibilità di prenotazione di ciascun libro.

PROPOSTE DI LAVORO

Realizzare lo schema concettuale con il diagramma ER e/o UML e stabilire i vincoli che devono essere rispettati.

Nota

Dopo aver completato il capitolo 5, costruire anche lo schema relazionale e query SQL di vario tipo.

1 ..... **Programmazione cinematografica**
In ogni cinema vengono proiettati uno o più film.
Di ogni cinema si conosce il nome, la città, l'indirizzo e il numero dei posti.
Di ogni film interessano il titolo, il genere (comico, drammatico ecc.), la durata, l'anno di produzione, il regista e gli attori.
Per ogni attore e regista sono noti nome, anno di nascita, nazionalità e genere abituale. Un attore può essere anche regista.

2 ..... **Scuola superiore**
Di ogni persona della scuola, studenti, personale docente e non docente vengono conservate le usuali informazioni anagrafiche.
Per ogni studente si deve conservare la storia scolastica, cioè le classi a cui lo studente è appartenuto e i risultati finali di ogni anno per ogni materia.
Il personale docente può essere di ruolo o supplente e può insegnare una o più materie.
Per quanto riguarda i locali della scuola, vi sono aule (associate alle classi), laboratori (associati alle materie) e uffici (associati in vario modo al personale).
Deve essere memorizzato anche l'orario scolastico per poter conoscere l'orario delle classi e degli insegnanti.

3 ..... **Conti correnti bancari**
Di ciascun cliente si conosce il codice fiscale, il nome e l'indirizzo. I clienti sono intestatari di uno o più conti.
Un conto è intestato ad uno o più clienti. Di ciascun conto si conoscono la data d'accensione, il tasso, le spese e il fido.
I clienti possono effettuare sui propri conti un numero imprecisato di movimenti; ogni movimento è riferito a un conto e modifica il saldo del conto. Di ciascun movimento si conosce il tipo (deposito, prelievo ecc.) la data e l'importo.

4 ..... **CD musicali**
Ogni CD ha un titolo e un'etichetta ed è composto da più brani, ognuno dei quali ha una durata ed è cantato da uno o più artisti o gruppi musicali; per ogni gruppo musicale devono essere noti i componenti del gruppo, per ogni artista il nome, il nome proprio se il nome con cui è noto è un nome d'arte, e note biografiche.

Loescher Editore - Vietata la vendita e la diffusione

5 ..... **Registro Automobilistico**

Di ogni veicolo (auto, camion o moto) interessano proprietario, targa, cilindrata e cavalli fiscali; delle auto si deve conoscere il numero di posti, dei camion il peso a vuoto.
Ogni veicolo appartiene a un unico proprietario ma a un proprietario possono appartenere più veicoli.
I proprietari possono essere persone fisiche (con codice fiscale, nome e indirizzo) o società (con codice fiscale, partita iva, ragione sociale, sede).

6 ..... **Ricette di cucina**

Le ricette sono suddivise in una sequenza ordinata di passi.
Ogni ricetta ha un codice, un titolo, un grado di difficoltà e un tempo di preparazione (che può essere ricavato dalla somma della durata dei singoli passi).
Per ogni ricetta è indicata la quantità di ogni ingrediente.
Ogni ingrediente ha un codice, un nome e una unità di misura (litri, grammi ecc.).
Ogni passo è rappresentato dalla lavorazione (cottura, impasto, assemblaggio ecc.), dall'insieme dei componenti (ingredienti di base o semilavorati di passi precedenti) e dal nuovo semilavorato ottenuto come risultato, e ha una propria durata.
Alcune ricette possono avere riferimenti ad altre ricette, corredati da una descrizione della relazione tra le ricette.

7 ..... **Piccola azienda commerciale**

L'azienda ha un catalogo dei prodotti venduti con codice, descrizione e prezzo che varia a seconda dei quantitativi. Per ogni prodotto è necessario conoscere l'attuale livello di disponibilità (quantità presente in azienda), i fornitori che lo mettono a disposizione, e a quale prezzo e il fornitore abituale.
Si vogliono mantenere le informazioni relative a tutti gli acquisti fatti (con articoli e fornitore) e a tutte le vendite (con articoli e cliente).
Dei fornitori si conosce la partita iva, il nome e l'indirizzo; deve essere noto il debito nei confronti di ciascuno (la somma dei totali delle fatture intestate dal fornitore); per ogni fornitore si mantiene anche il fido (il debito massimo concesso all'azienda dal fornitore).
Per ogni acquisto si memorizzano i dati delle fatture ricevute con il numero progressivo, la data, il totale, i dati relativi al fornitore e quelli relativi ad uno o più articoli acquistati.
I clienti registrati, di cui si conosce codice fiscale, nome e indirizzo, hanno uno sconto e una particolare modalità di pagamento (contanti, 30 gg., 90 gg., 30/60/90 gg.); deve essere noto il credito nei confronti di ciascuno (la somma dei totali delle fatture intestate al cliente); ogni cliente ha anche un fido (credito massimo concessogli).
Durante la vendita vengono emesse fatture che prevedono: numero progressivo, data, dati del cliente, elenco dei prodotti (con quantità, prezzo unitario dipendente dalla quantità, prezzo cumulativo per prodotto), totale fattura, sconto, imponibile, importo IVA.

8 ..... **Esposizione d'arte contemporanea**

Per ogni opera si devono memorizzare codice, autore e anno di realizzazione.
Ogni opera è un quadro o una scultura; per ogni quadro si vogliono memorizzare dimensioni e tecnica (olio, tempera ecc.), per ogni scultura altezza, peso e materiale.
Di ogni autore si conoscono codice fiscale, nome, anno di nascita, e uno o più numeri di telefono.
Ogni opera è esposta in una sala di un solo edificio; di ogni piano dell'edificio si conosce l'altezza dei soffitti; di ogni sala si conosce il piano a cui si trova, un numero identificativo e le dimensioni.
La sorveglianza delle sale è affidata a guardiani; di ogni guardiano si conoscono i dati personali, l'ora di inizio e fine del turno e le sale (una o più) di cui è responsabile.

Loescher Editore - Vietata la vendita e la diffusione

9 ..... Menu di un ristorante

Il menu comprende una lista di piatti; ciascun piatto appartiene ad una categoria (antipasti, primi, secondi ecc.) ed è caratterizzato da nome e prezzo; di ogni piatto si conoscono anche gli ingredienti.
Per ciascun piatto possono essere indicati uno o più vini consigliati per accompagnarlo.
I vini sono caratterizzati da nome, tipo (bianco o rosso), anno di produzione e prezzo per bottiglia.
Il menu comprende anche pasti a prezzo fisso (caratterizzati da nome e prezzo) che sono composti da alcuni piatti della lista, e possono permettere anche scelte tra due o più piatti alternativi.

10 ..... Gestori telefonici

Per ogni gestore di telefonia fissa o mobile si vogliono memorizzare codice, nome e possibili tariffe offerte ai clienti.
Ogni tariffa ha un codice specifico del gestore che la offre e il prezzo della telefonata per minuto.
Ogni persona (con codice fiscale, nome, residenza) può essere cliente di più gestori telefonici e può anche avere più contratti con lo stesso gestore, con tariffe diverse.
Ogni contratto per un certo numero telefonico riporta la tariffa relativa e la data in cui è stata attivata.
Si registrano le telefonate relative a ciascun contratto: chiamante, numero chiamato e durata, data di effettuazione.

11 ..... Prenotazioni ferroviarie

Ogni treno è identificato da un tipo (IC, EC ecc.) e da un numero, ed è caratterizzato da una stazione di partenza, una stazione di arrivo, un orario di partenza e un orario di arrivo. Alcuni treni viaggiano soltanto alcuni giorni (feriali, festivi ecc.).
Ogni treno è costituito da più carrozze, contrassegnate da un numero.
Ogni carrozza è di prima o di seconda classe e contiene più compartimenti di 6 posti ciascuno; ogni posto della carrozza è identificato da un numero progressivo.
Ogni treno percorre una o più tratte. Di ogni tratta si conoscono la stazione di partenza e la stazione di arrivo, la lunghezza e la tariffa in base al tipo di treno e alla classe della carrozza.
Ogni cliente (di cui sono noti nome e numero di telefono) può effettuare una prenotazione di più posti di una certa classe su un treno per più tratte. Per ogni prenotazione si vogliono memorizzare i posti assegnati al cliente e l'importo dovuto.

12 ..... Agenzia di viaggi

Ogni prenotazione di un viaggio comprende una combinazione di spostamenti e di soggiorni e prevede data di partenza, costo complessivo e registrazione dei dati anagrafici dei partecipanti.
Ogni spostamento è caratterizzato da località di partenza e di destinazione, mezzo di trasporto, tempo di percorrenza, data e ora di partenza e di arrivo, prezzo.
Ciascun soggiorno in una località, di cui è specificata la durata, avviene in un albergo il cui costo giornaliero dipende dal periodo dell'anno.

13 ..... Metadati

Si vogliono memorizzare i metadati relativi a un database relazionale.
Ogni relazione ha un nome e un insieme di attributi; uno o più attributi costituiscono la chiave.
Ogni attributo ha un nome (gli attributi di una relazione hanno nomi diversi ma in relazioni diverse possono esserci attributi con lo stesso nome) e un dominio.
Per ogni utente, di cui è registrato l'account, è indicato l'insieme delle relazioni a cui è consentito l'accesso.

Loescher Editore - Vietata la vendita e la diffusione

14 ..... Agenzia di collocamento.

Di ogni persona che cerca lavoro interessano oltre alle informazioni anagrafiche: la posizione attuale, le posizioni precedenti, i titoli di studio posseduti, le competenze.
Ogni posizione di lavoro è caratterizzata da datore di lavoro, tipo di attività, date di inizio e fine del rapporto.
Ogni titolo di studio è caratterizzato da scuola, data e voto.
Ogni competenza è caratterizzata da un nome e un livello di conoscenza.
Di ogni società che offre lavoro si vogliono memorizzare, oltre ai dati generali, le descrizioni dei profili del personale cercato; ogni profilo è caratterizzato da titoli di studio posseduti, competenze minime richieste, esperienze richieste (in base a tipo di attività e anni di esperienza).

15 ..... Compagnia assicurativa

Si vogliono memorizzare:
- per ogni cliente: codice fiscale, nome e indirizzo;
- per ogni auto: targa, modello, cilindrata, cavalli fiscali;
- per ogni incidente: l'auto assicurata coinvolta, la valutazione del danno e la percentuale di colpa.

Ogni cliente può avere più automobili e può stipulare più polizze.
Ogni polizza stipulata da un cliente per un'auto è caratterizzata dalle date di inizio e fine validità, dalla classe di rischio corrente e dal premio pagato.
Per il calcolo della classe di rischio e del premio si utilizza un tariffario annuale che in base ai cavalli fiscali e alla classe di rischio indica il premio corrispondente.
La classe di rischio è calcolata come classe dell'anno precedente meno uno più numero degli incidenti (rimane a 1 se era 1 e non ci sono stati incidenti).

16 ..... Censimento delle famiglie di una città

Si vogliono censire i nuclei familiari di una città.
Si vogliono memorizzare gli edifici, con gli appartamenti di cui sono composti.
Un nucleo familiare è costituito dalle persone che abitano in un appartamento.
Si vogliono descrivere le relazioni di parentela tra tutte le persone censite.

17 ..... Test a risposta multipla

Si vogliono memorizzare separatamente domande e risposte.
Ogni domanda è caratterizzata da un punteggio, un testo ed eventuali figure (anche più di una).
A ogni domanda sono associate più risposte, alcune corrette, altre no.
Ogni risposta è caratterizzata da un testo e da eventuali figure e può essere proposta come risposta a più domande, ma può essere corretta per alcune domanda ed errata per altre.
Si vogliono predisporre dei test che comprendono un certo numero di domande predefinite.

18 ..... Figure vettoriali

Si vogliono descrivere figure vettoriali bidimensionali indicando il nome e le figure geometriche (segmenti, rettangoli, circonferenze ecc.) che la compongono indicando per ciascuna di esse le informazioni necessarie a ridisegnarle: posizione e dimensioni (in base al tipo di figura), colore, riempimento ecc.
Si vuole anche registrare la storia delle modifiche apportate a ciascuna figura vettoriale.

19 ..... Una Olimpiade

Ogni nazione che partecipa ha una squadra composta da più atleti.
Ogni gara prevede più sessioni: varie eliminatorie, due semifinali e una finale.
Il calendario delle gare indica per ogni sessione data e orari di inizio e fine.
Per ogni sessione si desidera mantenere la graduatoria dei primi otto classificati.

Loescher Editore - Vietata la vendita e la diffusione

5 Il modello logico

Prerequisiti

- Saper utilizzare il modello concettuale per definire in modo formale le specifiche informali sulla realtà di interesse

Obiettivi

Conoscenze

- Fattori che influenzano le prestazioni di un database
- Ristrutturazione dello schema concettuale: tecniche per rimuovere le generalizzazioni, partizionamento e accorpamento delle entità, controllo della ridondanza, scelta degli identificatori principali
- Tecniche per tradurre: entità, associazioni uno a uno, uno a molti, molti a molti, ricorsive, ternarie
- Normalizzazione, dipendenza funzionale, forme normali: prima forma normale, seconda forma normale, terza forma normale, forma normale di Boyce-Codd

Abilità/capacità

- Ristrutturare modelli concettuali
- Tradurre un modello concettuale nel corrispondente modello relazionale
- Riconoscere in quale forma normale si trova uno schema di relazione
- Ridurre uno schema relazionale in seconda forma normale, terza forma normale, forma normale di Boyce-Codd

Loescher Editore - Vietata la vendita e la diffusione

5.1 LA PROGETTAZIONE LOGICA

Dallo schema concettuale (ER o UML) si deve ricavare uno **schema logico** composto dalle relazioni.

Prima della traduzione nello schema logico, può essere necessario **ristrutturare** il modello concettuale per semplificare la traduzione, poiché non tutti i costrutti del modello concettuale hanno una corrispondenza naturale nel modello logico, e per ottimizzare le prestazioni.

Le **prestazioni** si basano principalmente su due parametri:

- **costo di una operazione**, che dipende dal numero di istanze di entità e associazioni che devono essere visitate mediamente per effettuare una operazione (per esempio rispondere a una interrogazione);
- **occupazione di memoria** necessaria per memorizzare i dati.

Per studiare questi parametri è necessario conoscere il **volume dei dati**, cioè il numero di istanze di ogni entità e associazione e le dimensioni di ciascun attributo, e le **caratteristiche delle operazioni**, cioè in particolare quali sono i dati (entità e associazioni) coinvolti e la frequenza delle operazioni.

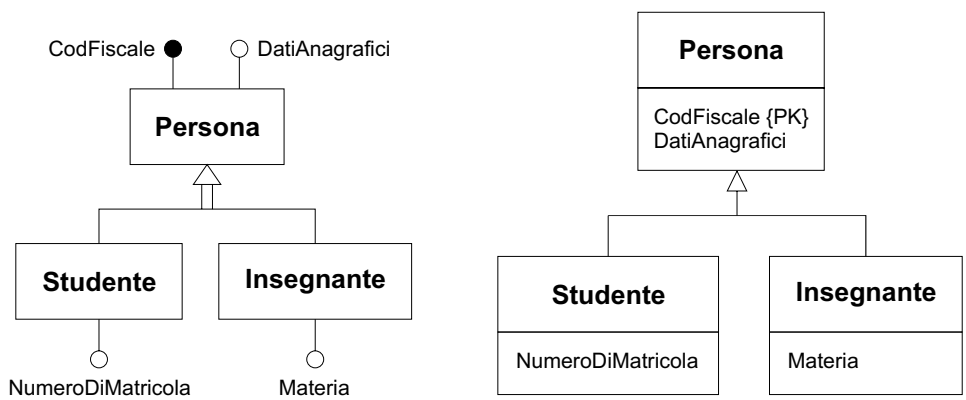
5.2 RISTRUTTURAZIONE DELLO SCHEMA CONCETTUALE

Eliminazione delle generalizzazioni

I sistemi di gestione di basi di dati di solito non consentono di rappresentare direttamente la generalizzazione.

Esempio 5.1

La generalizzazione *Persona* - *Studente*, *Insegnante* non può essere rappresentata direttamente.



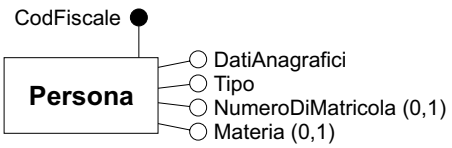
Le **generalizzazioni** possono essere eliminate tramite:

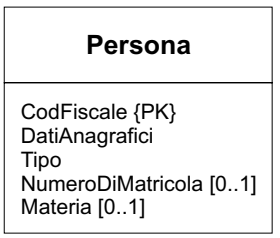
- **accorpamento delle figlie nel padre**: le entità figlie vengono accorpate nell'entità padre che ha gli attributi e le associazioni di tutte le figlie e un attributo per distinguerle; c'è uno spreco di memoria per la presenza di valori *null* ma è richiesto un minor numero di accessi;

Loescher Editore - Vietata la vendita e la diffusione

Esempio 5.2

Studente e *Insegnante* si possono accorpare in *Persona*, aggiungendo a *Persona* un attributo *Tipo* per distinguerle, più gli attributi di entrambe: *NumeroDiMatricola* e *Materia*, ognuno impostato o no in base al valore di *Tipo*.

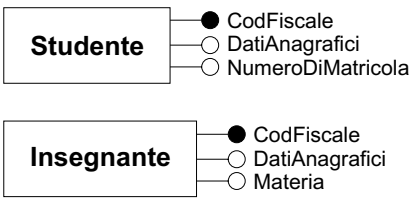


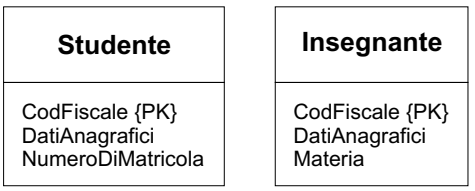


- **accorpamento del padre nelle figlie**: gli attributi e le associazioni dell'entità padre vengono duplicati nelle entità figlie; questo metodo è possibile solo se la generalizzazione è totale, altrimenti alcune istanze dell'entità padre non potrebbero essere rappresentate; si ha un risparmio di memoria perché non ci sono valori *null* e anche in questo caso si ha una riduzione degli accessi;

Esempio 5.3

Se per la realtà di interesse la generalizzazione di *Persona* in *Studente* e *Insegnante* è totale si possono inserire i dati anagrafici di *Persona* in *Studente* e in *Insegnante*.

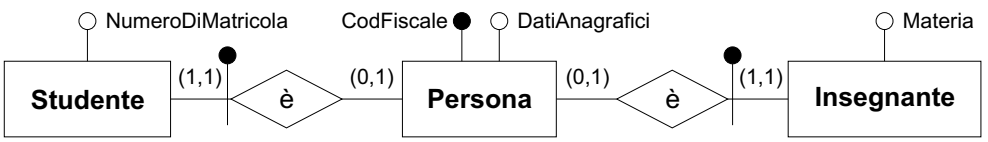


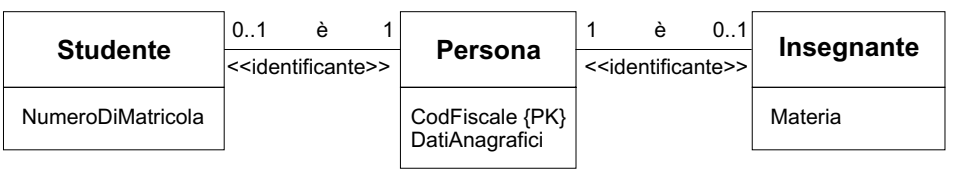


- **sostituzione** della generalizzazione con **associazioni** uno a uno che legano l'entità padre con ciascuna delle entità figlie (la partecipazione dell'entità padre all'associazione è opzionale); questo metodo è conveniente quando la generalizzazione non è totale e ci sono operazioni che fanno riferimento separatamente ai dati del padre o a quelli di una delle figlie; si ha un risparmio di memoria ma un incremento degli accessi.

Esempio 5.4

Si può sostituire la generalizzazione di *Persona* in *Studente* e *Insegnante* con le associazioni uno a uno tra *Persona* e *Studente* e tra *Persona* e *Insegnante* (la partecipazione di *Persona* alle associazioni è opzionale).





Loescher Editore - Vietata la vendita e la diffusione

Partizionamento ed accorpamento di entità

Per ridurre il numero degli accessi può essere utile **separare** attributi di uno stesso concetto che vengono utilizzati da operazioni diverse e **raggruppare** attributi di concetti diversi utilizzati dalle stesse operazioni.
Raggruppando attributi di concetti diverse potrebbero però esserci valori *null*; l'accorpamento si fa di solito per associazioni di tipo uno a uno, mai per molti a molti, raramente per uno a molti perché possono generare ridondanze che poi devono essere risolte con la normalizzazione.

Esempio 5.5

Può essere utile partizionare l'entità *Studente* in due entità: una che contiene i dati anagrafici e l'altra che contiene i dati scolastici, se i dati sono gestiti da applicazioni o persone diverse.

Può essere utile accorpare le entità *Persona* e *DiplomaScolastico* se è sempre necessario conoscere i dati del diploma di una persona (si suppone che ne abbia uno solo); potrebbero però esserci persone senza diploma e quindi valori *null*.

Nota

Anche le associazioni possono essere partizionate o accorpate; si può decomporre una associazione tra due entità in due o più associazioni tra le stesse entità o viceversa.

Ridondanze

Nel modello concettuale ci possono essere varie forme di **ridondanza**, tra cui:

- **attributi derivabilli** da altri attributi della stessa entità, istanza per istanza;

L'attributo *DensitàDiPopolazione* di *Nazione* è derivabile dagli attributi *Popolazione* e *Superficie* (*DensitàDiPopolazione* = *Popolazione* / *Superficie*).

- attributi derivabili da attributi di altre entità o associazioni attraverso **funzioni aggregative o conteggi**.

Esempio 5.6

La *Popolazione* di un *Continente* può essere calcolata sommando la *Popolazione* delle nazioni del continente.
La *Popolazione* di una *Città* può essere ricavata dalla associazione *Residenza* tra *Persona* e *Città*, contando le persone che risiedono nella città.

La presenza di dati derivati riduce gli accessi necessari per il calcolo dei dati, ma richiede una maggiore occupazione di memoria (anche se in genere trascurabile) e la necessità di operazioni aggiuntive per mantenere il dato derivato aggiornato, garantendone la consistenza.
La decisione di inserire in un'entità dati derivati va presa caso per caso; in alcuni casi la ridondanza è indispensabile.

Esempio 5.7

Date le tra entità *Prodotto* con il *Prezzo* di listino, *Cliente* con i dati anagrafici e *Acquisto* che indica la *Quantità* acquistata di un prodotto, è fondamentale inserire in *Acquisto* anche il *Prezzo* al momento dell'ordine, perché poi il prezzo del prodotto potrebbe subire delle modifiche.

Loescher Editore - Vietata la vendita e la diffusione

Scelta degli identificatori principali

La scelta degli identificatori principali (**chiavi**) è essenziale perché nel modello relazionale le chiavi vengono usate per stabilire legami tra dati in relazioni diverse.
Inoltre sulle chiavi primarie vengono costruiti automaticamente indici che rendono più efficiente reperire i dati.

Criteri di decisione per la scelta quando sono possibili più identificatori

È preferibile utilizzare:

- un identificatore con pochi attributi che riduce le dimensioni degli indici con risparmio di memoria;
- un identificatore che viene utilizzato da molte operazioni.

Nota

L'identificatore principale non può contenere valori *null* perché i valori *null* non permettono l'accesso a tutte le istanze.

5.3 DALLO SCHEMA CONCETTUALE ALLO SCHEMA RELAZIONALE

Dopo aver progettato lo schema concettuale, entità e associazioni devono essere tradotti in **relazioni** che costituiscono lo **schema logico** (nel modello relazionale si parla anche di **schema relazionale**).

Rimangono valide le **regole aziendali** definite nella progettazione.
Bisogna aggiungere alla documentazione i **vincoli di integrità referenziale** introdotti nella traduzione.

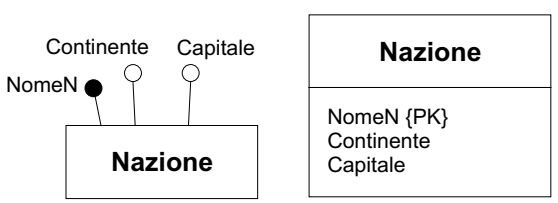
Entità e attributi

Ogni **entità** è rappresentata da una relazione con gli stessi attributi; gli attributi derivati, cioè che si ottengono con elaborazioni, di solito non vanno indicati. La chiave è l'identificatore principale.
Il nome dell'entità diventa il nome della relazione.

Nota

Nella relazione il nome può essere usato al plurale per maggior chiarezza e per ridurre la possibilità di omonimie con i nomi degli attributi, soprattutto quelli che fungono da chiavi esterne.

Esempio 5.8

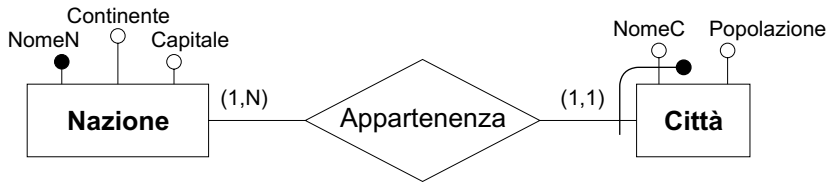


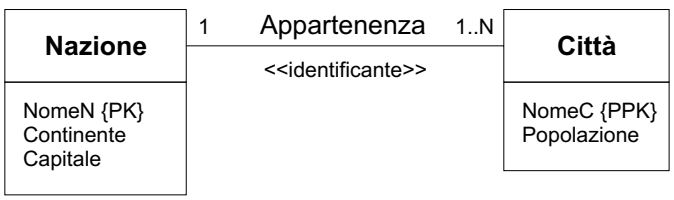
Nazioni (<u>NomeN</u>, Continente, Capitale)

Le entità con un **identificatore esterno** vengono realizzate inserendo nella relazione anche l'identificatore dell'entità "identificante".

Loescher Editore - Vietata la vendita e la diffusione

Esempio 5.9





Nazioni (NomeN, Continente, Capitale)
Città (NomeC, Nazione, Popolazione)
Città.Nazione è in integrità referenziale con *Nazioni.NomeN*.

Associazioni

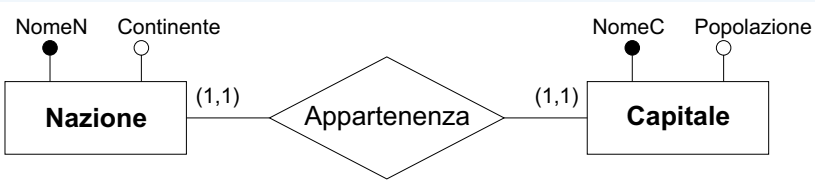
In generale le associazioni uno a uno e uno a molti vengono realizzate aggiungendo una chiave esterna a una delle relazioni, le associazioni molti a molti aggiungendo una nuova relazione.
Gli identificatori possono essere ridenominati per maggiore chiarezza; la ridenominazione è essenziale in caso di associazioni ricorsive.

Associazioni 1:1

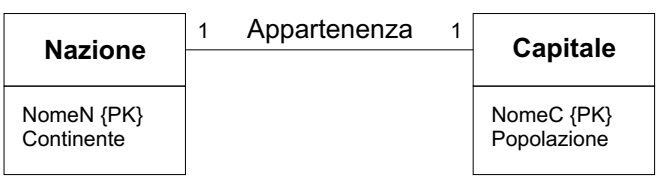
Di solito per realizzare associazioni uno a uno si aggiunge una **chiave esterna** a una delle relazioni.

Se la partecipazione di entrambe le entità è **obbligatoria** è indifferente a quale relazione aggiungere la chiave esterna; insieme alla chiave esterna vengono aggiunti anche gli eventuali attributi dell'associazione.
In alcuni casi la chiave esterna potrebbe essere aggiunta ad entrambe le relazioni per semplificare e rendere più efficienti le interrogazioni. Gli attributi dell'associazione però vengono aggiunti a una sola delle due relazioni.

Esempio 5.10



In alternativa:



Nazioni (NomeN, Continente, Capitale)
Capitali (NomeC, Popolazione)
Nazioni.Capitale è in integrità referenziale con *Capitali.NomeC*.

Loescher Editore - Vietata la vendita e la diffusione

Nazioni (NomeN, Continente)
Capitali (NomeC, Popolazione, Nazione)
Capitali.Nazione è in integrità referenziale con *Nazioni.NomeN*.

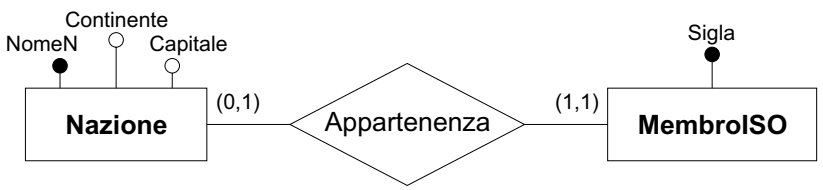
Nazioni (NomeN, Continente, Capitale)
Capitali (NomeC, Popolazione, Nazione)
Nazioni.Capitale è in integrità referenziale con *Capitali.NomeC*;
Capitali.Nazione è in integrità referenziale con *Nazioni.NomeN*.

Nota

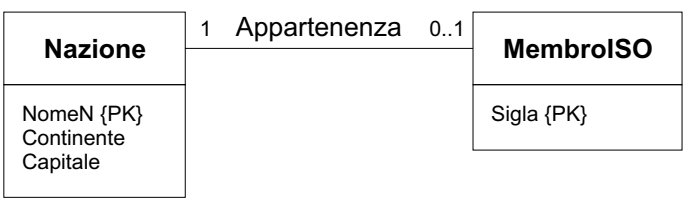
Si potrebbe pensare di usare un'unica relazione che contiene gli attributi delle due entità ma tale soluzione è da scartare perché al momento della traduzione il modello concettuale è già stato ristrutturato con precise scelte su accorpamento e partizionamento.

Se una delle due entità partecipa in modo **facoltativo** bisogna scegliere l'entità a cui aggiungere la chiave esterna in modo da non avere valori *null* in questo campo: in ER all'entità che non ha cardinalità inferiore 0, in UML all'entità che ha molteplicità inferiore 0.

Esempio 5.11

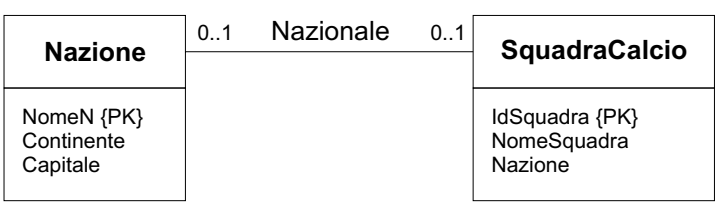


Nazioni (NomeN, Continente, Capitale)
MembriISO (Sigla, Nazione)
MembriISO.Nazione è in integrità referenziale con *Nazioni.NomeN*.



Se entrambe le entità partecipano in modo **facoltativo** si aggiunge una nuova relazione che ha come chiavi esterne le chiavi di entrambe le entità; la nuova relazione ha anche gli attributi dell'associazione; la chiave della nuova relazione può essere una qualsiasi delle due chiavi esterne.

Esempio 5.12



Nazioni (NomeN, Continente, Capitale)
SquadreDiCalcio (IdSquadra, NomeSquadra, Nazione)

Loescher Editore - Vietata la vendita e la diffusione

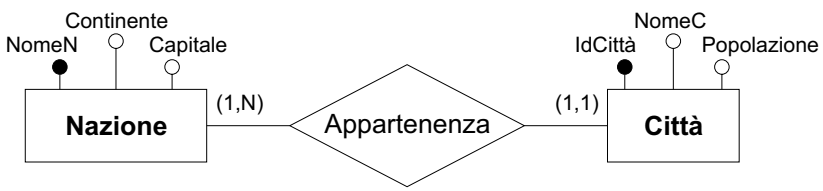
Nazionale (Nazione, Squadra)
SquadreDiCalcio.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Nazionale.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Nazionale.Squadra è in integrità referenziale con *SquadreDiCalcio.IdSquadra*.

Associazioni 1:N

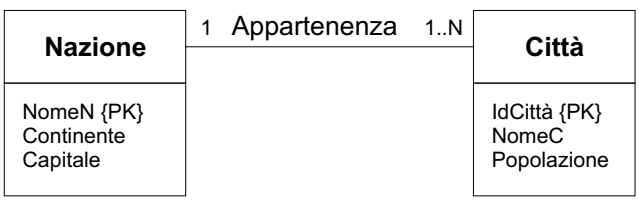
Per realizzare associazioni uno a molti si aggiunge una **chiave esterna** alla relazione corrispondente:
- in ER all'entità con cardinalità 1, per fare riferimento all'entità con cardinalità N;
- in UML all'entità con molteplicità N, per fare riferimento all'entità con molteplicità 1;

insieme alla chiave esterna vengono aggiunti anche gli eventuali attributi dell'associazione.

Esempio 5.13

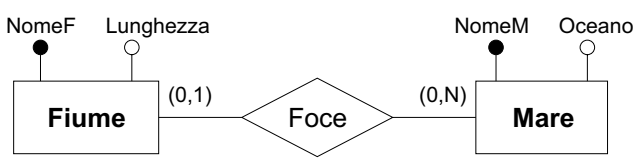


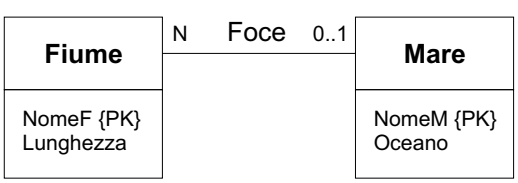
Nazioni (NomeN, Continente, Capitale)
Città (IdCittà, NomeC, Nazione, Popolazione)
Città.Nazione è in integrità referenziale con *Nazioni.NomeN*.



La partecipazione **facoltativa** dell'entità dal lato uno può provocare la presenza di valori *null* nella chiave esterna (che si può segnalare aggiungendo un asterisco dopo il nome dell'attributo). Per evitare la presenza di valori *null* nella chiave esterna, si può realizzare l'associazione aggiungendo una nuova relazione.

Esempio 5.14





Fiumi (NomeF, Lunghezza, Mare*)
Mari (NomeM, Oceano)
Fiumi.Mare è in integrità referenziale con *Mari.NomeM*; non tutti i fiumi sfociano in un mare (possono essere affluenti di un altro fiume o essere immissari di un lago) quindi *Mare* può avere valori *null* (è segnalato dall'asterisco).

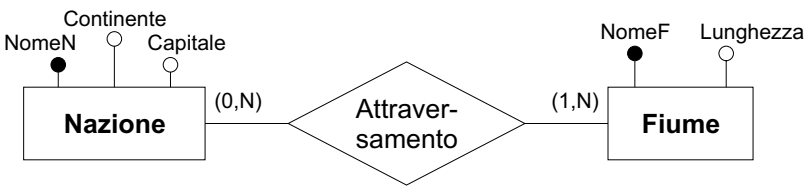
Fiumi (NomeF, Lunghezza)
Mari (NomeM, Oceano)
Foce (Fiume, Mare)
Foce.Fiume è in integrità referenziale con *Fiumi.NomeF*;
Foce.Mare è in integrità referenziale con *Mari.NomeM*.

Loescher Editore - Vietata la vendita e la diffusione

Associazioni N:N

Per realizzare una associazione N:N si deve aggiungere una nuova relazione che ha come attributi gli identificatori delle entità che partecipano all'associazione e gli eventuali attributi dell'associazione. La chiave della relazione è costituita dagli identificatori delle entità che partecipano all'associazione e dagli attributi dell'associazione necessari a garantire l'unicità delle tuple nella relazione ottenuta.

Esempio 5.15



Nazioni (<u>NomeN</u>, Continente, Capitale)
Fiumi (<u>NomeF</u>, Lunghezza)
Attraversamento (<u>Nazione</u>, <u>Fiume</u>) oppure **Attraversamento** (<u>Fiume</u>, <u>Nazione</u>)
Attraversamento.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Attraversamento.Fiume è in integrità referenziale con *Fiumi.NomeF*.

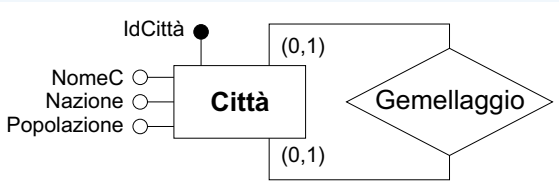
La scelta dell'ordine degli identificatori delle entità nella nuova relazione dipende dalla frequenza delle operazioni di ricerca dei fiumi di una nazione o delle nazioni attraversate da un fiume.

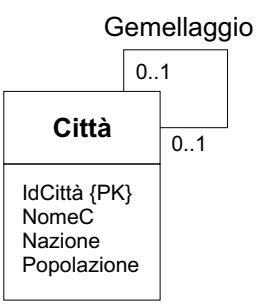
Associazioni ricorsive

Le associazioni ricorsive sono casi particolari delle situazioni già viste.

Se l'associazione ricorsiva è uno a uno si può aggiungere una chiave esterna a una delle relazioni o creare una nuova relazione.

Esempio 5.16





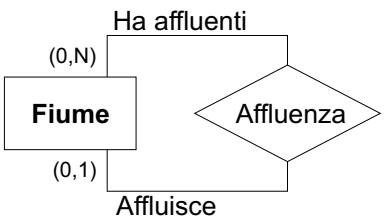
Città (<u>IdCittà</u>, NomeC, Nazione, Popolazione, Gemella*)
Città.Gemella è in integrità referenziale con *Città.IdCittà*.
Però, dato che la partecipazione di *Città* a *Gemellaggio* è facoltativa, è meglio aggiungere una nuova relazione:
Città (<u>IdCittà</u>, NomeC, Nazione, Popolazione)
Gemellaggio (<u>Città1</u>, Città2)
Gemellaggio.Città1 è in integrità referenziale con *Città.IdCittà*;
Gemellaggio.Città2 è in integrità referenziale con *Città.IdCittà*.

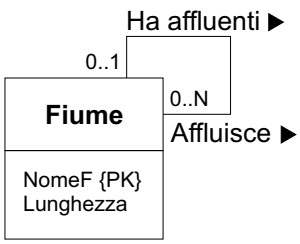
Loescher Editore - Vietata la vendita e la diffusione

Se l'associazione ricorsiva è uno a molti di solito si aggiunge una chiave esterna alla relazione relativa all'entità che partecipa all'associazione con cardinalità 1 in ER e molteplicità N in UML.

Per evitare la presenza di valori *null* nella chiave esterna si può realizzare l'associazione aggiungendo una nuova relazione.

Esempio 5.17



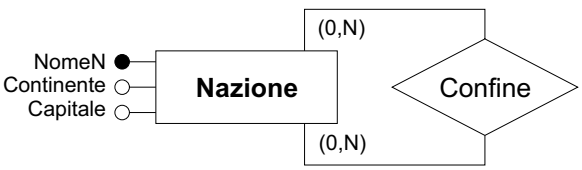


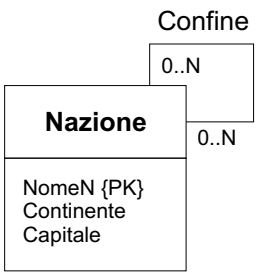
Fiumi (NomeF, Lunghezza, AffluenteDi*)
Fiumi.AffluenteDi è in integrità referenziale con *Fiumi.NomeF*.

Fiumi (NomeF, lunghezza)
Affluenza (Affluente, Fiume)
Affluenza.Affluente è in integrità referenziale con *Fiumi.NomeF*;
Affluenza.Fiume è in integrità referenziale con *Fiumi.NomeF*.

Se l'associazione ricorsiva è molti a molti si deve creare una nuova relazione.

Esempio 5.18





Nazioni (NomeN, Continente, Capitale)
Confine (Nazione1, Nazione2)
Confine.Nazione1 è in integrità referenziale con *Nazioni.NomeN*;
Confine.Nazione2 è in integrità referenziale con *Nazioni.NomeN*.

Associazioni ternarie

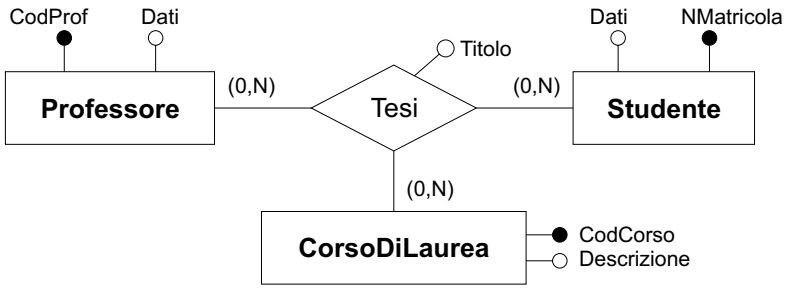
Le associazioni ternarie di solito sono di tipo molti a molti e si realizzano in modo analogo alle associazioni molti a molti: dopo aver creato una relazione per ogni entità si aggiunge una relazione che ha come attributi gli identificatori delle entità che partecipano all'associazione e gli eventuali attributi dell'associazione.
La chiave della relazione è costituita dagli identificatori delle entità che partecipano all'associazione e dagli attributi dell'associazione necessari a garantire l'unicità delle tuple nella relazione ottenuta.

Nota

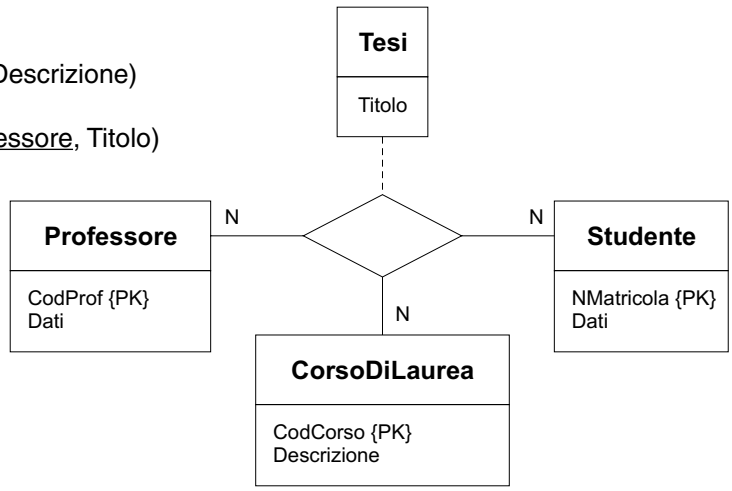
Se una delle entità partecipa alla relazione ternaria con cardinalità/molteplicità massima 1, l'associazione può essere tradotta con una relazione che contiene come chiavi esterne gli identificatori delle altre entità.

Loescher Editore - Vietata la vendita e la diffusione

Esempio 5.19



Studenti (NMatricola, Dati)
CorsiDiLaurea (CodCorso, Descrizione)
Professori (CodProf, Dati)
Tesi (Studente, Corso, Professore, Titolo)
Tesi.Studente è in integrità referenziale con *Studenti.NMatricola*;
Tesi.Corso è in integrità referenziale con *CorsiDiLaurea.CodCorso*;
Tesi.Professore è in integrità referenziale con *Professori.CodProf*.



Riassumendo

Ogni **entità** diventa una relazione.

Ogni **attributo** di un'entità diventa un attributo di una relazione.

L'**identificatore principale** di una entità diventa la chiave primaria della relazione.

Le **entità con un identificatore esterno** vengono realizzate inserendo nella relazione anche l'identificatore dell'entità "identificante".

Associazioni uno a uno:

- se entrambe le entità hanno partecipazione obbligatoria una delle due entità ha la chiave esterna dell'altra;
- se una delle due entità partecipa in modo facoltativo, la chiave esterna va aggiunta in ER all'entità che non ha cardinalità inferiore 0, in UML all'entità che ha molteplicità inferiore 0;
- se entrambe le entità partecipano in modo facoltativo si aggiunge una nuova relazione con le due chiavi esterne.

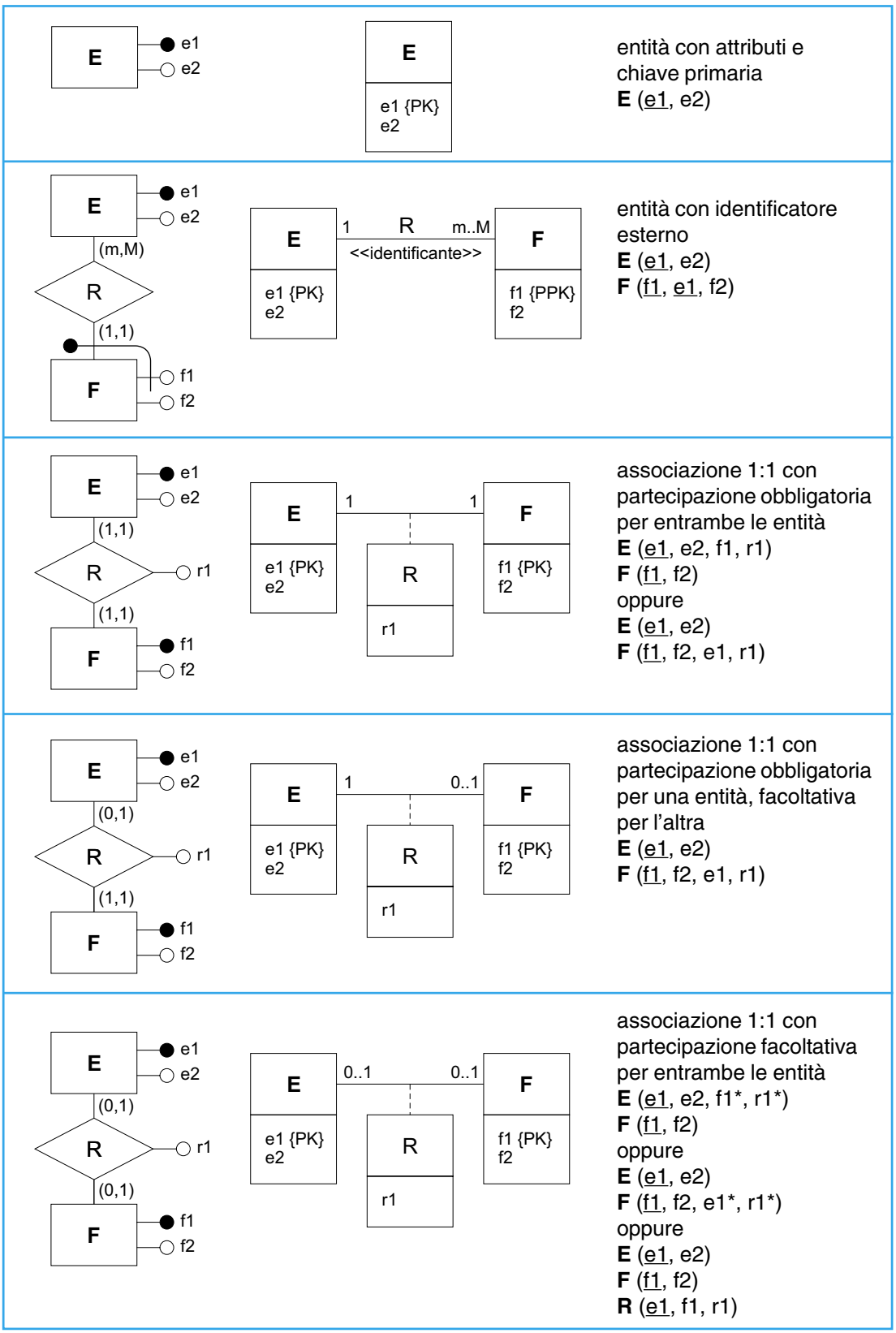
Associazioni uno a molti:

- la relazione corrispondente all'entità in ER con cardinalità 1, in UML con molteplicità N, contiene la chiave esterna dell'altra;
- se l'entità dal lato uno partecipa in modo facoltativo è meglio aggiungere una nuova relazione.

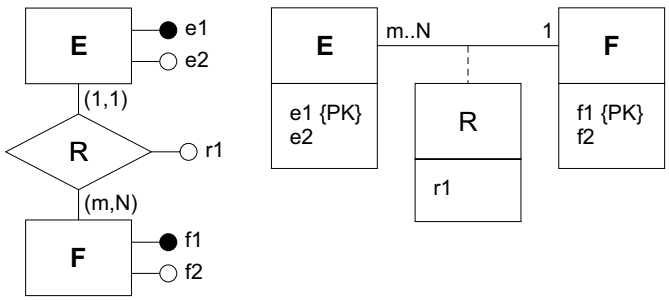
Associazioni molti a molti: si aggiunge una nuova relazione.

Loescher Editore - Vietata la vendita e la diffusione

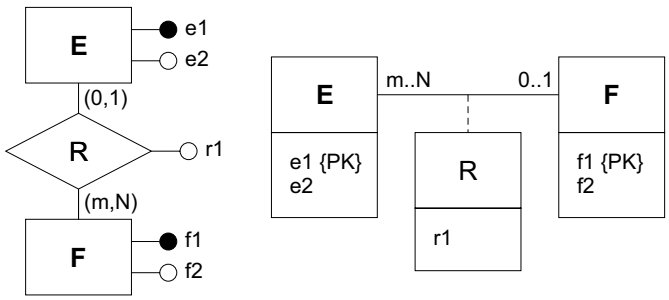
Figura 5.1 *Schemi riassuntivi*



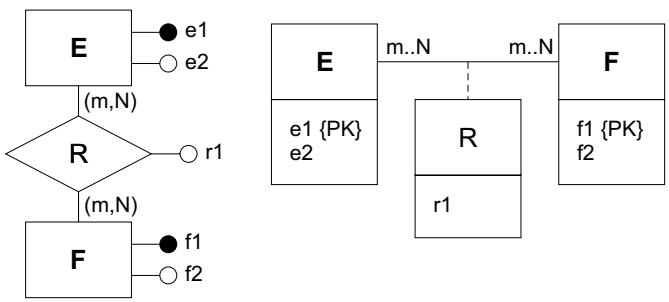
Loescher Editore - Vietata la vendita e la diffusione



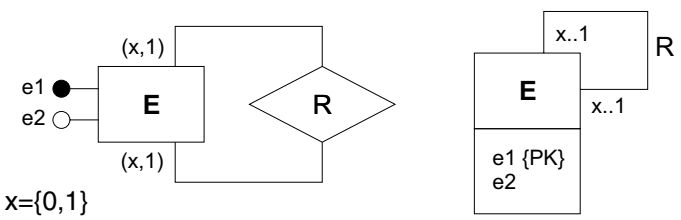
associazione 1:N con partecipazione obbligatoria
E (e1, e2, f1, r1)
F (f1, f2)



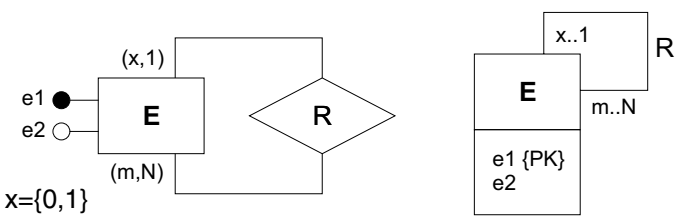
associazione 1:N con partecipazione facoltativa dell'entità dal lato 1
E (e1, e2)
F (f1, f2)
R (e1, f1, r1)
oppure
E (e1, e2, f1*, r1*)
F (f1, f2)



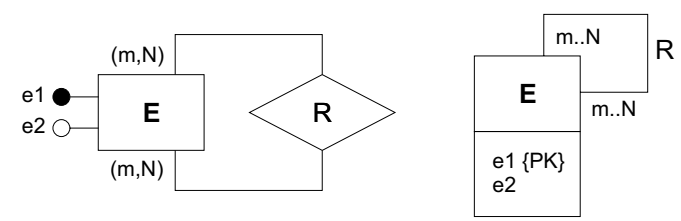
associazione N:N
E (e1, e2)
F (f1, f2)
R (e1, f1, r1)



associazione ricorsiva 1:1
E (e1a, e2, e1b*)
oppure
E (e1, e2)
R (e1a, e1b)

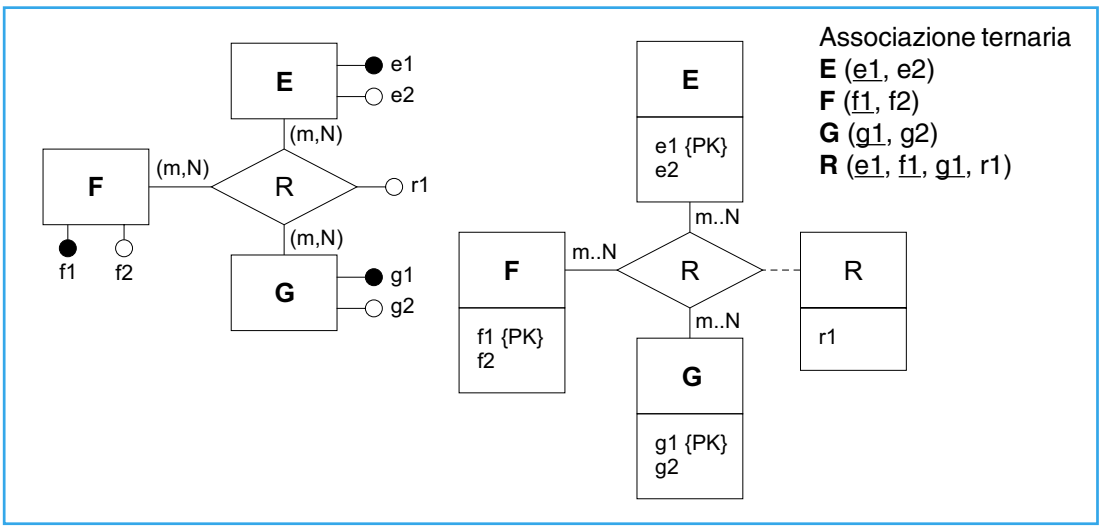


associazione ricorsiva 1:N
E (e1a, e2, e1b*)
oppure
E (e1, e2)
R (e1a, e1b)



associazione ricorsiva N:N
E (e1, e2)
R (e1a, e1b)

Loescher Editore - Vietata la vendita e la diffusione



5.4 ESEMPI SVOLTI

Traduzione dello schema concettuale in schema relazionale per gli esempi 4.38, 4.39 e 4.40.

ESEMPIO 5.20

Geografia (esempio 4.38)

Nazioni (NomeN, Continente, Capitale, Superficie, Popolazione)
Continenti (Sigla, NomeCon)
Città (IdCittà, NomeC, Nazione, Popolazione)
Mari (NomeM, Oceano)
Fiumi (NomeF, Lunghezza)
Laghi (NomeL, Superficie)
Litorale (Nazione, Mare)
Attraversamento (Nazione, Fiume)
Presenza (Nazione, Lago)
Confine (Nazione1, Nazione2)
Affluenza (Affluente, Fiume)
Immissario (Fiume, Lago)
Foce (Fiume, Mare)

Nazioni.Continente è in integrità referenziale con *Continenti.Sigla*;
Nazioni.Capitale è in integrità referenziale con *Città.IdCittà*;
Città.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Litorale.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Litorale.Mare è in integrità referenziale con *Mari.NomeM*;
Attraversamento.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Attraversamento.Fiume è in integrità referenziale con *Fiumi.NomeF*;
Presenza.Nazione è in integrità referenziale con *Nazioni.NomeN*;
Presenza.Lago è in integrità referenziale con *Laghi.NomeL*;
Confine.Nazione1 è in integrità referenziale con *Nazioni.NomeN*;
Confine.Nazione2 è in integrità referenziale con *Nazioni.NomeN*;

Loescher Editore - Vietata la vendita e la diffusione

Affluenza.Affluente è in integrità referenziale con *Fiumi.NomeF*;
Affluenza.Fiume è in integrità referenziale con *Fiumi.NomeF*;
Immissario.Fiume è in integrità referenziale con *Fiumi.NomeF*;
Immissario.Lago è in integrità referenziale con *Laghi.NomeL*;
Foce.Fiume è in integrità referenziale con *Fiumi.NomeF*;
Foce.Mare è in integrità referenziale con *Mari.NomeM*.

Esempio 5.21

Biblioteca (esempio 4.39)

Libri (Codice, Titolo, Editore, AnnoPubblic)
Editori (IdEditore, Nome, Sede, Telefono, AnnoFondazione)
Autori (IdAutore, Nome, Indirizzo, Telefono)
Scrittura (Autore, Libro)
Utenti (IdUtente, Nome, Indirizzo, Telefono)
Prestiti (Utente, Libro, DataPrestito, DataRestituzione)

Libri.Editore è in integrità referenziale con *Editori.IdEditore*;
Scrittura.Autore è in integrità referenziale con *Autori.IdAutore*;
Scrittura.Libro è in integrità referenziale con *Libri.Codice*;
Prestiti.Utente è in integrità referenziale con *Utenti.IdUtente*;
Prestiti.Libro è in integrità referenziale con *Libri.Codice*.

Esempio 5.22

Gestione aziendale (esempio 4.40)

Sedi (CodSede, Citta, Prov, Via, Civico)
Settori (CodSettore, Nome)
Dipendenti (CodFisc, Nome, Cognome, DataDiNascita, Telefono, Stipendio, InizioRapporto, FineRapporto, FineContratto, Settore, Sede, Responsabile)
Articoli (Codice, Modello, Descrizione, PrezzoNetto, Settore, Produttore)
Produttori (Nome, NomeRappr, TelRappr)
Scontrini (Numero, Data, SedeScontr, Articolo, Quantita)
ArticoliSedi (Articolo, Sede, PezziMagazzino, PezziEsposti, QtaMinima)

Dipendenti.Settore è in integrità referenziale con *Settori.CodSettore*;
Dipendenti.Sede è in integrità referenziale con *Sedi.CodSede*;
Dipendenti.Responsabile è in integrità referenziale con *Dipendenti.CodFisc*;
Articoli.Settore è in integrità referenziale con *Settori.CodSettore*;
Articoli.Produttore è in integrità referenziale con *Produttori.Nome*;
Scontrini.SedeScontr è in integrità referenziale con *Sedi.CodSede*;
Scontrini.Articolo è in integrità referenziale con *Articoli.Codice*;
ArticoliSedi.Articolo è in integrità referenziale con *Articoli.Codice*;
ArticoliSedi.Sede è in integrità referenziale con *Sedi.CodSede*.

Per i *Settori* viene aggiunto un codice come chiave artificiale, più conveniente poiché i settori sono riferiti sia dai dipendenti che dagli articoli.
Per la generalizzazione dei dipendenti viene utilizzata un'unica relazione, considerando dipendenti a tempo determinato tutti i dipendenti con il campo *FineContratto* diverso da *null*.
La relazione *Scontrini* raccoglie i dati delle associazioni tra *Scontrini* e *Articoli* e tra *Scontrini* e *Sedi*.

Loescher Editore - Vietata la vendita e la diffusione

5.5 LA NORMALIZZAZIONE

Se nella stessa relazione sono presenti più concetti in realtà indipendenti tra loro ci possono essere ridondanze e anomalie di inserimento, cancellazione o aggiornamento di dati.
In questo caso è utile procedere alla normalizzazione delle relazioni, decomponendo la relazione di partenza in due o più relazioni, una per ogni concetto indipendente.

La **normalizzazione** è un procedimento che ha lo scopo di separare i concetti indipendenti in relazioni diverse, uno per ogni relazione.

La normalizzazione diminuisce la ridondanza dei dati e quindi i problemi di inconsistenza, ma complica le operazioni di accesso ai dati perché richiede operazioni di join tra le tabelle per ottenere le stesse informazioni. Integrità e consistenza sono però obiettivi prioritari rispetto alla velocità di accesso.

Esistono vari livelli di normalizzazione (**forme normali**), cioè sono definiti diversi gradi di forme normali in cui possono trovarsi le relazioni.

Nota

Non è sempre possibile raggiungere i livelli più alti di normalizzazione perché in alcuni casi il procedimento potrebbe comportare una perdita di informazioni e quindi non essere applicabile.

I livelli di normalizzazione certificano la qualità dello schema logico del database.
La normalizzazione in realtà è necessaria se il progetto non è stato realizzato correttamente.
Stabilire il livello di normalizzazione delle relazioni serve quindi a valutare la bontà dello schema logico e ad evidenziare eventuali problemi, che potrebbero richiedere anche la revisione della progettazione concettuale.

ANOMALIE DI INSERIMENTO, CANCELLAZIONE E AGGIORNAMENTO DATI

Le **anomalie di inserimento** riguardano l'impossibilità di inserire alcuni dati se non vengono inseriti anche altri dati correlati.

Le **anomalie di cancellazione** si verificano quando alcuni dati vengono eliminati in modo indesiderato per effetto della cancellazione di altri dati correlati.

Le **anomalie di aggiornamento** si verificano quando nel database sono presenti delle ridondanze e non tutte le occorrenze dello stesso dato vengono aggiornate.

ESEMPIO 5.23

Tabella non normalizzata con anomalie di inserimento, cancellazione e aggiornamento.

Nazioni

Città	Nazione	PopolazioneCittà	PopolazioneNazione
Roma	Italia	2.642.700	57.884.000
Berlino	Germania	3.387000	82.504.000
Torino	Italia	903.400	57.884.000
Parigi	Francia	9.319.000	60.561.000
Vienna	Austria	1.608.000	8.118.000
Lione	Francia	413.000	60.561.000
Venezia	Italia	276.100	57.884.000
...	...	...	...

Loescher Editore - Vietata la vendita e la diffusione

La tabella comprende i due concetti di città e nazione in realtà indipendenti.

La popolazione della nazione è ridondante e viene ripetuta per ogni città della stessa nazione.
La ridondanza, oltre a sprecare spazio su disco, è causa delle anomalie già citate:

- anomalia di inserimento: non è possibile inserire la popolazione di una nazione se non si inserisce nessuna città di quella nazione;
- anomalia di cancellazione: se si eliminano tutte le città di una nazione si perde anche il dato relativo alla popolazione della nazione;
- anomalia di aggiornamento: se la popolazione di una nazione cambia, bisogna modificare il dato in tutte le righe relative alle città di quella nazione; se qualche riga non viene aggiornata lo stesso dato riporta due valori diversi e non è più possibile individuare quale sia corretto.

Per eliminare le ridondanze ed evitare le anomalie bisogna separare i dati indipendenti in due diverse tabelle:

Città

Città	Nazione	PopolazioneCittà
Roma	Italia	2.642.700
Berlino	Germania	3.387000
Torino	Italia	903.400
Parigi	Francia	9.319.000
Vienna	Austria	1.608.000
Lione	Francia	413.000
Venezia	Italia	276.100
...	...	...

Nazioni

Nazione	PopolazioneNazione
Italia	57.884.000
Germania	82.504.000
Francia	60.561.000
Austria	8.118.000
...	...

Dipendenze funzionali

La **dipendenza funzionale** è un particolare vincolo di integrità per il modello relazionale che descrive legami di tipo funzionale tra gli attributi di una relazione.

Se X e Y sono due sottoinsiemi di attributi non vuoti di una relazione, si dice che Y dipende funzionalmente da X , e si scrive **X → Y** , se Y è determinato in modo univoco dai valori degli attributi di X. Si dice anche che X è un determinante per Y.

In pratica esiste una funzione che associa ad ogni valore di X un unico valore di Y.
Se due tuple *t1* e *t2* di hanno gli stessi valori sugli attributi di X, hanno valori uguali anche sugli attributi di Y (X determina Y).

Se un insieme di attributi è **chiave candidata** di una relazione allora è un determinante per ogni attributo della relazione; se un insieme di attributi è determinante per ogni attributo della relazione allora è una superchiave.

Si dice che tra due insiemi A e C di attributi esiste una **dipendenza transitiva**, o che C dipende transitivamente da A, quando A determina un insieme di attributi B e B determina C.

Le **ridondanze** e le **anomalie** sono causate dalle dipendenze funzionali X → Y che permettono la presenza di più tuple fra loro uguali sugli attributi in X; in altre parole sono causate dalle dipendenze funzionali X → Y tali che **X non contiene una chiave**.

Loescher Editore - Vietata la vendita e la diffusione

La normalizzazione è la semplificazione delle dipendenze funzionali per eliminare le dipendenze funzionali fra gli attributi non chiave di una relazione e quindi per evitare che si verifichino anomalie.

Prima forma normale

La prima forma normale è la forma in cui si trovano tutte le relazioni: tutti gli attributi sono di tipo elementare, senza ulteriori restrizioni.

Una relazione è in **prima forma normale** se rispetta i requisiti fondamentali del modello relazionale:

- tutte le tuple hanno lo stesso numero e tipo di attributi;
- i valori di un attributo sono dello stesso tipo cioè appartengono allo stesso dominio;
- le tuple sono tutte diverse tra loro;
- l'ordine delle tuple è irrilevante;
- gli attributi sono di tipo elementare cioè non possono essere scomposti in attributi più semplici e non possono contenere attributi ripetuti.

Se un'entità ha attributi composti basta sostituirli con i singoli attributi componenti; se un'entità ha attributi multipli bisogna utilizzare un'altra entità legata alla prima da una associazione.

Nota

Un esempio di modellazione di attributi multivalore è stato presentato nel paragrafo 4.6.

Seconda forma normale

La seconda forma normale riguarda il caso di relazioni con chiavi composte da più attributi ed elimina la dipendenza parziale degli attributi dalla chiave.

Una relazione è in **seconda forma normale** se tutti gli attributi non chiave dipendono funzionalmente da tutta la chiave, e non in modo parziale; cioè se si toglie un qualsiasi attributo dalla chiave, gli altri attributi non sono più determinati in modo univoco.

Una relazione in prima forma normale si può trasformare in uno schema in seconda forma normale creando delle relazioni separate per gli attributi che dipendono solo da una parte della chiave.

Esempio 5.24

R (A, B , C, D)

chiave:
{A, B}

dipendenze funzionali:
{A, B} → C
A → D

R non è in seconda forma normale perché *D* non dipende dall'intera chiave.

Lo schema normalizzato diventa:
R1 (A, B, C)
R2 (A, D)

Loescher Editore - Vietata la vendita e la diffusione

Esempio 5.25

R (Città, Nazione, PopolazioneCittà, PopolazioneNazione)

chiave:
{Città, Nazione}
(la chiave è formata da {*Città*, *Nazione*} perché il nome della città da solo non è univoco a causa di omonimie)

dipendenze funzionali:
{Città, Nazione} → PopolazioneCittà
Nazione → PopolazioneNazione

R non è in seconda forma normale, infatti l'attributo *PopolazioneNazione* non dipende da tutta la chiave ma solo da *Nazione*.

Lo schema normalizzato diventa:
R1 (Città, Nazione, PopolazioneCittà)
R2 (Nazione PopolazioneNazione)

Terza forma normale

La terza forma normale elimina le dipendenze funzionali tra attributi che non appartengono alla chiave.

Nota

Nelle relazioni in seconda forma normale ci possono essere anomalie perché ci possono essere dipendenze funzionali fra attributi non chiave.

Una relazione è in **terza forma normale** (**3NF** - 3 Normal Form) se per ogni dipendenza funzionale X → Y definita su di essa è verificata almeno una delle seguenti condizioni:

- X è una superchiave (cioè contiene una chiave);
- ogni attributo di Y è contenuto in almeno una chiave della relazione.

Nota

Nella definizione le dipendenze funzionali si intendono non banali. Una dipendenza X → Y si dice banale se qualche attributo di Y compare tra gli attributi di X (per esempio {*Città*, *Nazione*} → *Nazione*.

Se una relazione ha una sola chiave candidata si può dire più semplicemente che è in terza forma normale se è in seconda forma normale e non esistono attributi che dipendono transitivamente dalla chiave (cioè non ci sono attributi non chiave che dipendono da altri attributi non chiave).

Nota

La terza forma normale elimina la dipendenza transitiva degli attributi dalla chiave.

Teorema

Ogni relazione può sempre essere portata in terza forma normale.

Una relazione in seconda forma normale si può trasformare in uno schema in terza forma normale creando delle relazioni separate per gli attributi di Y che non appartengono a nessuna chiave (e che quindi dipendono transitivamente dalla chiave).

Loescher Editore - Vietata la vendita e la diffusione

Esempio 5.26

R (A, B , C, D)

chiave candidata:
A

dipendenze funzionali:
A → B
A → C
B → D

R non è in terza forma normale perché *D* dipende transitivamente dalla chiave tramite l'attributo *B* (in *B* → *D*, *B* non è una chiave e *D* non appartiene a nessuna chiave).

Lo schema normalizzato diventa:
R1 (A, B, C)
R2 (B, D)

Esempio 5.27

R (Nazione, Continente, PopolazioneContinente)

chiave candidata:
Nazione

dipendenze funzionali:
Nazione → Continente
Continente → PopolazioneContinente

R non è in terza forma normale infatti:
Nazione → *Continente* soddisfa la prima condizione perché *Nazione* è una chiave, ma:
Continente → *PopolazioneContinente* non soddisfa nessuna delle due condizioni: *Continente* non è una chiave e *PopolazioneContinente* non appartiene a nessuna chiave.

Si può anche dire che *R* non è in terza forma normale perché *PopolazioneContinente* dipende da *Continente* che non è un campo chiave.

Lo schema normalizzato diventa:
R1 (Nazione, Continente)
R2 (Continente, PopolazioneContinente)

Dopo la terza forma normale ci sono ulteriori livelli di normalizzazione ma non sempre è possibile arrivarci.
Spesso si può considerare accettabile uno schema in terza forma normale.

Forma normale di Boyce-Codd

Una relazione è in forma normale di Boyce-Codd se tutti i suoi determinanti sono chiavi candidate, cioè se ogni insieme di attributi dal quale dipendono altri attributi può svolgere la funzione di chiave.

Una relazione è in forma normale di **Boyce-Codd** (**BCNF** - Boyce-Codd Normal Form) se per ogni dipendenza funzionale X → Y vale la seguente condizione:

- X è una superchiave (cioè contiene una chiave).

Loescher Editore - Vietata la vendita e la diffusione

Una relazione si può trasformare in BCNF individuando le dipendenze funzionali, raggruppando quelle con la parte determinante (sinistra) uguale, e proiettando la relazione non normalizzata sull'insieme di attributi corrispondente a ciascun gruppo. Vengono prodotte tante relazioni quante sono le dipendenze funzionali con diverso primo membro.

Anomalie e dipendenze non si presentano in relazioni in BCNF perché i concetti indipendenti sono separati, uno per relazione.

Esempio 5.28

R (A, B, C)

chiave candidata:
A

dipendenze funzionali:
A → B soddisfa la condizione perché *A* è una chiave;
B → C non soddisfa la condizione perché *B* non è una chiave;

R non è in BCNF.

Lo schema normalizzato diventa:
R1 (A, B)
R2 (B, C)

Esempio 5.29

R (Nazione, Continente, SuperficieContinente)

chiave candidata:
Nazione

dipendenze funzionali:
Nazione → Continente soddisfa la condizione perché *Nazione* è una chiave;
Continente → SuperficieContinente non soddisfa la condizione perché *Continente* non è una chiave;

R non è in BCNF.

Lo schema normalizzato diventa:
R1 (Nazione, Continente)
R2 (Continente, SuperficieContinente)

Esempio 5.30

R (A, B, C, D)

chiavi candidate:
A
D

dipendenze funzionali:
A → B soddisfa la condizione perché *A* è una chiave candidata;
B → C non soddisfa la condizione perché *B* non è una chiave candidata;
A → D soddisfa la condizione perché *A* è una chiave candidata;

Loescher Editore - Vietata la vendita e la diffusione

D → A soddisfa la condizione perché *D* è una chiave candidata;
D → B soddisfa la condizione perché *D* è una chiave candidata;

R non è in BCNF.

Lo schema normalizzato diventa:
R1 (A, B, D)
R2 (B, C)

Esempio 5.31

R (Nazione, Continente, SuperficieContinente, SiglaNazione)

chiavi candidate:
Nazione
SiglaNazione

dipendenze funzionali:

Nazione → Continente	soddisfa la condizione perché *Nazione* è una chiave candidata;
Continente → SuperficieContinente	non soddisfa la condizione perché *Continente* non è una chiave candidata;
Nazione → SiglaNazione	soddisfa la condizione perché *Nazione* è una chiave candidata;
SiglaNazione → Nazione	soddisfa la condizione perché *SiglaNazione* è una chiave candidata;
SiglaNazione → Continente	soddisfa la condizione perché *SiglaNazione* è una chiave candidata;

R non è in BCNF.

Lo schema normalizzato diventa:
R1 (Nazione, Continente, SiglaNazione)
R2 (Continente, SuperficieContinente)

Non sempre è possibile raggiungere la BCNF.
Anche quando è possibile, spesso l'insieme delle dipendenze funzionali definite su una relazione può essere complesso e può essere difficile individuare le dipendenze funzionali su cui si deve basare la decomposizione.

Proprietà delle decomposizioni

Una decomposizione deve sempre essere senza perdita di informazioni.

Una decomposizione è **senza perdita di informazioni** (Lossless Join property) se è sempre possibile ricostruire la relazione originaria (cioè ottenere le stesse informazioni) mediante join delle relazioni ottenute dalla scomposizione.

Se la decomposizione non soddisfa questa proprietà, la relazione risultante dal join contiene anche tuple aggiuntive (o spurie) prive di significato.

Condizione sufficiente perché sia possibile decomporre una relazione R in due relazioni R1 e R2 senza perdita di informazioni è che l'insieme degli attributi comuni ad R1 e R2 sia una superchiave per almeno una delle due relazioni decomposte.

Loescher Editore - Vietata la vendita e la diffusione

Esempio 5.32

Progetti sviluppati in una sola scuola.

R (Progetto, Coordinatore, Scuola)

chiave candidata:
Progetto

dipendenze funzionali:

Progetto → Coordinatore	ciascun progetto ha un coordinatore;
Coordinatore → Scuola	ciascun coordinatore insegna presso un'unica scuola;
Progetto → Scuola	(transitiva dalle altre due) ciascun progetto è sviluppato presso un'unica scuola.

Ciascun coordinatore può partecipare a più progetti ma solo nella sua scuola.

R non è in BCNF perché *Coordinatore* non è chiave candidata.

Dividendo in:
R1 (Coordinatore, Scuola)
R2 (Progetto, Scuola)
non si ottiene una relazione equivalente: applicando il join si ottengono anche tuple che non facevano parte della relazione originaria.
La decomposizione non è senza perdita di informazioni, infatti non è più possibile individuare i coordinatori dei progetti.
La condizione che l'insieme degli attributi comuni ad *R1* e *R2* sia una superchiave per almeno una delle due relazioni decomposte non è soddisfatta infatti l'attributo comune *Scuola* non è chiave di nessuna delle due relazioni (non è primo membro di alcuna dipendenza funzionale).

Sfruttando la dipendenza *Coordinatore* → *Scuola* si può ottenere la decomposizione:
R1 (Coordinatore, Scuola)
R2 (Progetto, Coordinatore)
Il join produce la relazione di partenza infatti, l'attributo comune *Coordinatore* è chiave per la prima relazione.

Una buona decomposizione deve garantire anche la **conservazione delle dipendenze**.

> Una decomposizione conserva le dipendenze se tutte le dipendenze definite sulla relazione originaria sono verificabili direttamente anche sulle relazioni ottenute nella scomposizione.

La conservazione delle dipendenze garantisce il mantenimento dei vincoli di integrità originari. Garantire che siano rappresentati gli stessi vincoli di integrità permette di rilevare aggiornamenti illeciti.

Esempio 5.33

R (Allievo, Materia, Insegnante)
si suppone che un insegnante possa insegnare una sola materia.

chiavi candidate:
{Allievo, Materia}
{Allievo, Insegnante}

dipendenze funzionali:
{Allievo, Materia} → Insegnante
Insegnante → Materia

Loescher Editore - Vietata la vendita e la diffusione

R non è in BCNF perché *Insegnante* non è chiave candidata.

Mediante proiezioni si ottiene:
R1 (Allievo, Insegnante)
R2 (Insegnante, Materia)

Si perde però la dipendenza:
{Allievo, Materia} → Insegnante
quindi sarebbe possibile inserire due insegnanti della stessa materia per lo stesso allievo senza che questo aggiornamento sia rilevato come illecito.

Relazione tra la BCNF e la 3NF

Se una relazione è in BCNF è anche in terza forma normale.

Nota

Gli schemi normalizzati ottenuti negli esempi da 5.28 a 5.31 quindi oltre che in BCNF sono anche in terza forma normale.

Non è vero però il viceversa: una relazione in terza forma normale può non essere in BCNF.

Non è possibile garantire sempre il raggiungimento della BCNF.
La terza forma normale invece è sempre raggiungibile.

Nota

Una relazione è in terza forma normale ma non in BCNF se c'è un attributo che appartiene a una chiave candidata che dipende da attributi non chiave: la 3NF richiede che ogni attributo che non appartiene a chiavi candidate dipenda da una superchiave, la BCNF lo richiede per ogni attributo (anche quelli che appartengono alla chiave).

La terza forma normale è meno forte della BCNF, presenta ridondanze ed è affetta da anomalie di cancellazione (cioè è possibile perdere informazioni a seguito di un'operazione di cancellazione) ma è sempre ottenibile.

Nota

Ridondanze e anomalie vengono tollerate nei casi in cui non è possibile effettuare una decomposizione che elimini la ridondanza e conservi le dipendenze.

La relazione *R (Allievo, Materia, Insegnante)* dell'esempio 5.33 non è decomponibile in relazioni in BCNF che conservino le dipendenze, però è in terza forma normale.
La relazione però ha una ridondanza: la materia di un insegnante è ripetuta su più tuple.
Inoltre se si cancellano tutte le tuple degli allievi che hanno un dato insegnante si perde l'informazione relativa alla materia di tale insegnante.

Esempio 5.34

Progetti sviluppati in più scuole.

R (Progetto, Scuola, Coordinatore)
ogni progetto ha più coordinatori ognuno in una scuola diversa; ogni coordinatore può essere coordinatore di più progetti però per ogni scuola un progetto ha un solo coordinatore; ogni coordinatore insegna presso un'unica scuola.

Chiavi candidate:
{Progetto, Scuola}
{Progetto, Coordinatore}

Loescher Editore - Vietata la vendita e la diffusione

Dipendenze funzionali:
{Progetto, Scuola} → Coordinatore
Coordinatore → Scuola

R non è in BCNF però è in 3NF perché:

{Progetto, Scuola} → Coordinatore	ha come primo membro una chiave;
Coordinatore → Scuola	*Coordinatore* non è chiave candidata ma *Scuola* fa parte della chiave.

Non è possibile decomporre la relazione *R* in BCNF in modo da conservare le dipendenze perché {*Progetto*, *Scuola*} → *Coordinatore* coinvolge tutti gli attributi.

Nella relazione in terza forma normale c'è una ridondanza: la scuola è ripetuta ogni volta che compare il coordinatore.

VERIFICA

Questionario

1. Che cos'è lo schema logico o schema relazionale di un database?
2. Perché può essere necessario ristrutturare lo schema concettuale?
3. Come si possono eliminare le generalizzazioni?
4. Quando può essere utile partizionare o accorpare le entità?
5. Quando è accettabile qualche forma di ridondanza?
6. Quali sono i criteri per la scelta dell'identificatore principale di ogni entità?
7. Come si traducono nello schema relazionale le entità?
8. Come si traducono nello schema relazionale le associazioni 1:1?
9. Come si traducono nello schema relazionale le associazioni 1:N?
10. Come si traducono nello schema relazionale le associazioni N:N?
11. Come si traducono nello schema relazionale le associazioni ricorsive?
12. Come si traducono nello schema relazionale le associazioni ternarie?
13. Che cos'è la normalizzazione?
14. Che cosa sono le forme normali?
15. Che cos'è una dipendenza funzionale?
16. Da che cosa sono causate ridondanze e anomalie?
17. Quando una relazione è in prima forma normale?
18. Quando una relazione è in seconda forma normale?
19. Quando una relazione è in terza forma normale?
20. Quando una relazione è in BCNF (forma normale di Boyce Codd)?

Test

1 ..... *Indicare se le seguenti affermazioni sono vere o false.*

V	F	
☐	☐	1) Lo schema logico o relazionale si rappresenta con i diagrammi ER o UML.
☐	☐	2) Il volume dei dati dipende dal numero di istanze di ogni entità e dalla dimensione di ogni attributo.
☐	☐	3) Una generalizzazione deve sempre essere ristrutturata.

Loescher Editore - Vietata la vendita e la diffusione

V	F	
☐	☐	4) La presenza di dati ridondanti nel database può causare inconsistenze.
☐	☐	5) Se sono possibili più identificatori è indifferente quale scegliere.
☐	☐	6) L'identificatore principale non può contenere valori *null*.
☐	☐	7) La normalizzazione riduce il numero di operazioni di join necessarie per effettuare le interrogazioni.
☐	☐	8) La normalizzazione riduce la ridondanza dei dati.
☐	☐	9) Se lo schema concettuale è stato realizzato correttamente la normalizzazione di solito non serve.
☐	☐	10) Un insieme di attributi è una chiave candidata se e solo se è un determinante per ogni attributo della relazione.
☐	☐	11) Si dice che esiste una dipendenza transitiva tra A e B quando A -> B e B -> A.
☐	☐	12) Una relazione può non essere in prima forma normale.
☐	☐	13) In una relazione in seconda forma può esserci un attributo che dipende transitivamente dalla chiave.
☐	☐	14) Ogni relazione può sempre essere portata in terza forma normale.
☐	☐	15) Ogni relazione può sempre essere portata in BCNF.
☐	☐	16) È accettabile la perdita di informazioni in una relazione pur di raggiungere la BCNF.
☐	☐	17) Perché sia possibile decomporre una relazione in due relazioni senza perdita di informazioni bisogna che le due nuove relazioni abbiano la stessa chiave.
☐	☐	18) Se trasformando una relazione in BCNF non si conservano le dipendenze si rendono possibili aggiornamenti illeciti.
☐	☐	19) Una relazione in terza forma normale è sempre anche in BCNF.
☐	☐	20) Una relazione in BCNF è sempre anche in terza forma normale.

2 ..... *Segnare tutte le affermazioni esatte.*

1) Il diagramma ER:

può essere tradotto nel modello relazionale come:

- ☐ Classi(<u>NomeC</u>, NumStudenti); Aule(<u>IdAula</u>, NumPosti);
- ☐ Classi(<u>NomeC</u>, NumStudenti, IdAula); Aule(<u>IdAula</u>, NumPosti, NomeC);
- ☐ Classi(NomeC, NumStudenti, <u>IdAula</u>); Aule(<u>IdAula</u>, NumPosti, NomeC);
- ☐ Classi(<u>NomeC</u>, NumStudenti, IdAula); Aule(<u>IdAula</u>, NumPosti);
- ☐ Classi(<u>NomeC</u>, NumStudenti); Aule(<u>IdAula</u>, NumPosti, NomeC);
- ☐ Classi(<u>NomeC</u>, NumStudenti); Aule(<u>IdAula</u>, NumPosti); Assegnazione(<u>IdAula</u>, NomeC);
- ☐ Classi(<u>NomeC</u>, NumStudenti); Aule(<u>IdAula</u>, NumPosti); Assegnazione(<u>NomeC</u>, IdAula);
- ☐ Classi(<u>NomeC</u>, NumStudenti); Aule(<u>IdAula</u>, NumPosti); Assegnazione(<u>NomeC</u>, <u>IdAula</u>).

Loescher Editore - Vietata la vendita e la diffusione

2) Il diagramma ER:

può essere tradotto nel modello relazionale come:

- Classi(NomeC, NumStudenti); Aule(IdAula, NumPosti);
- Classi(NomeC, NumStudenti, IdAula); Aule(IdAula, NumPosti, NomeC);
- Classi(NomeC, NumStudenti, IdAula); Aule(IdAula, NumPosti, NomeC);
- Classi(NomeC, NumStudenti, IdAula); Aule(IdAula, NumPosti);
- Classi(NomeC, NumStudenti); Aule(IdAula, NumPosti, NomeC);
- Classi(NomeC, NumStudenti); Aule(IdAula, NumPosti); Assegnazione(IdAula, NomeC);
- Classi(NomeC, NumStudenti); Aule(IdAula, NumPosti); Assegnazione(NomeC, IdAula);
- Classi(NomeC, NumStudenti); Aule(IdAula, NumPosti); Assegnazione(NomeC, IdAula).

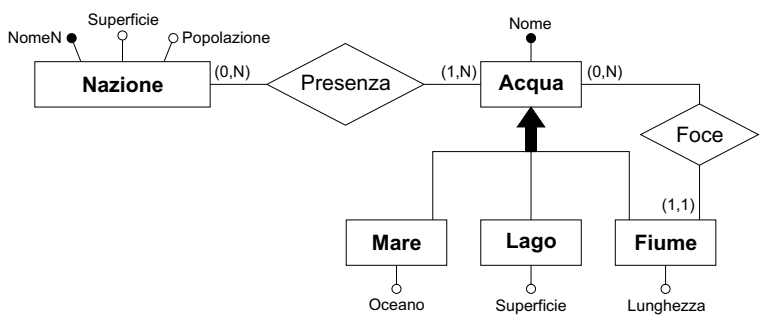
3) Il diagramma ER:

può essere tradotto nel modello relazionale come:

- Classi(NomeC, NumStudenti); Aule(IdAula, NumPosti);
- Classi(NomeC, NumStudenti, IdAula); Aule(IdAula, NumPosti, NomeC);
- Classi(NomeC, NumStudenti, IdAula); Aule(IdAula, NumPosti, NomeC);
- Classi(NomeC, NumStudenti, IdAula); Aule(IdAula, NumPosti);
- Classi(NomeC, NumStudenti); Aule(IdAula, NumPosti, NomeC);
- Classi(NomeC, NumStudenti); Aule(IdAula, NumPosti); Assegnazione(IdAula, NomeC);
- Classi(NomeC, NumStudenti); Aule(IdAula, NumPosti); Assegnazione(NomeC, IdAula);
- Classi(NomeC, NumStudenti); Aule(IdAula, NumPosti); Assegnazione(NomeC, IdAula).

Esercizi

1 ..... Ristrutturare la generalizzazione rappresentata nello schema ER:



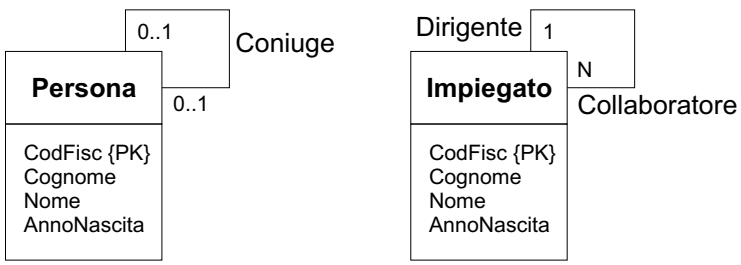
Loescher Editore - Vietata la vendita e la diffusione

2 ..... Individuare l'attributo ridondante per le seguenti relazioni e stabilire se è conveniente mantenerlo o meno:
1) **Fatture** (ImportoLordo, IVA, Importo netto);
2) **Studenti** (Matricola, Nome, Classe); **Classi** (NomeC, Aula, NumeroStudenti).

3 ..... Tradurre nel modello relazionale i diagrammi UML:

1)

2)



4 ..... Data la relazione
Studenti (Matricola, CodFiscale, Nome, ComuneResidenza, ProvinciaResidenza)
stabilire in che forma normale è (supponendo che non ci siano omonimie tra i comuni) ed eventualmente normalizzarla.

5 ..... Data la relazione
ComuniItaliani (Codice, Nome, Provincia, Regione)
stabilire in che forma normale è ed eventualmente normalizzarla.

6 ..... Data la relazione
Libri (Codice, Titolo, Autore, emailAutore, Editore, emailEditore, AnnoEdizione, Prezzo)
stabilire in che forma normale è ed eventualmente normalizzarla.

7 ..... Data la relazione **R** (A, B, C, D, E) e le dipendenze funzionali
A -> B,
{B, C} -> D
{D, E} -> A,
determinare le chiavi candidate di R e specificare se R è in 3NF o in BCNF.

8 ..... Data la relazione **R** (A, B, C, D, E) e le dipendenze funzionali
B -> C,
{C, D} -> E
{E, A} -> B,
determinare le chiavi candidate di R e specificare se R è in 3NF o in BCNF.

9 ..... Data la relazione **R** (A, B, C, D, E)
stabilire se le seguenti decomposizioni sono senza perdita di informazione:
1) **R1** (A, B, C, D)
   R2 (D, E)
2) **R1** (A, B, C, D)
   R2 (D, E)

10 ..... Data la relazione
Tesi (MatricolaStudente, NomeStudente, NomeRelatore, Dipartimento, FacoltaDocente, Materia, Argomento, NomeCorrelatore, TipoTesi)

Loescher Editore - Vietata la vendita e la diffusione

con le ipotesi:
- relatore e correlatore sono docenti; un docente appartiene a un solo dipartimento e a una sola facoltà ma può insegnare più materie;
- un docente può fare da correlatore per le tesi di un altro docente solo per una materia;
- un argomento di tesi è specifico di una materia;
- una materia può essere insegnata da più docenti;

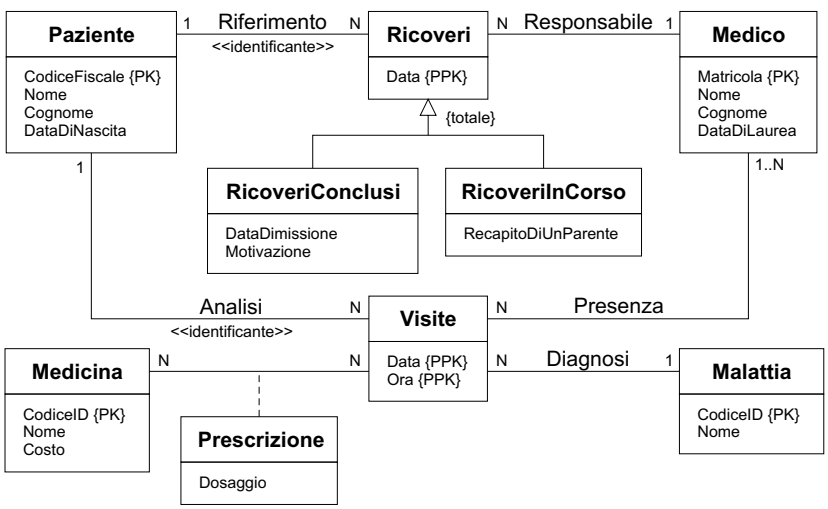
determinare le dipendenze funzionali e decomporla in BCNF o in 3NF.

11 ..... Data la relazione
GiocatoriCalcio (Cod, Cognome, Nome, CodRuolo, Ruolo, CodNaz, Nazione, DataNascita, Presenze)
determinare le dipendenze funzionali e decomporla in BCNF o in 3NF.

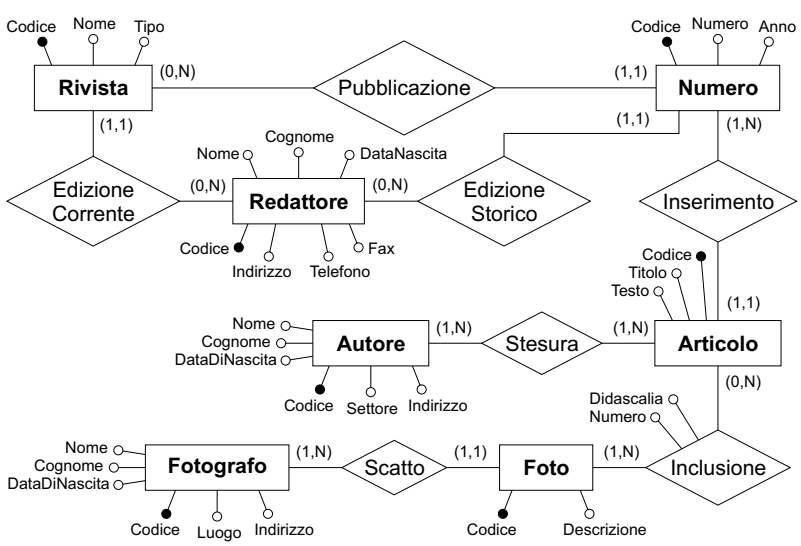
PROPOSTE DI LAVORO

Tradurre nello schema relazionale i seguenti diagrammi ER e UML. Costruire poi query SQL.

1 ..... **Reparto ospedaliero**.

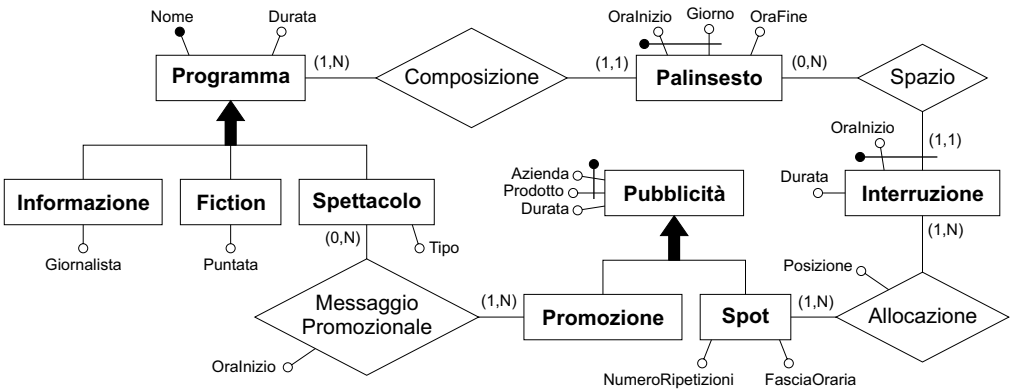


2 ..... **Riviste**.

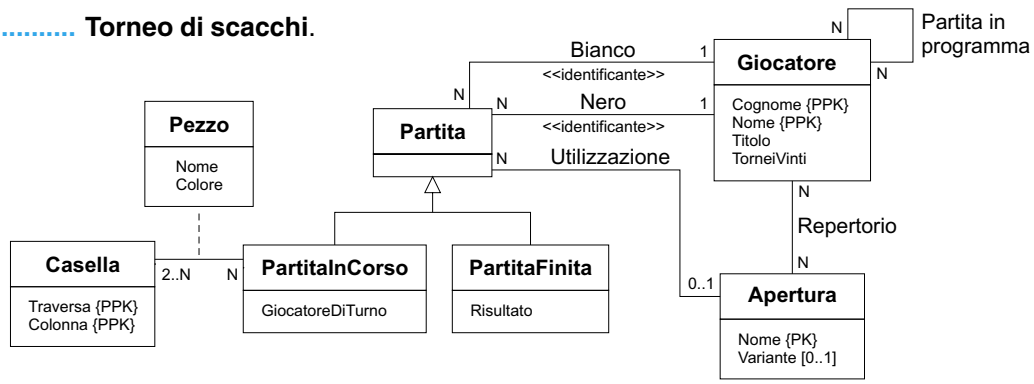


Loescher Editore - Vietata la vendita e la diffusione

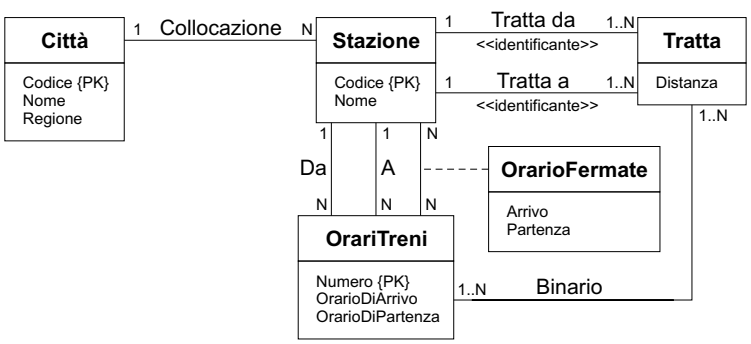
3 ..... Trasmissioni di una rete televisiva.



4 ..... Torneo di scacchi.



5 ..... Orario ferroviario.



Dati i seguenti schemi relazionali effettuare il reverse engineering (disegnare il diagramma ER o UML relativo al modello concettuale).

6 ..... **Impiegati** (Codice, Nome, Stipendio, Dipartimento)
Dipartimenti (NomeD, Città, Telefono, Direttore)
Progetti (NomeP, Budget, DataConsegna)
Partecipazione (Impiegato, Progetto, DataInizio)
Impiegati.Dipartimento è in integrità referenziale con *Dipartimenti.NomeD*;
Dipartimenti.Direttore è in integrità referenziale con *Impiegati.Codice*;
Partecipazione.Impiegato è in integrità referenziale con *Impiegati.Codice*;
Partecipazione.Progetto è in integrità referenziale con *Progetti.NomeP*.

Loescher Editore - Vietata la vendita e la diffusione

7 ..... **Classi** (IdC, Classe, Sezione)
Studenti (IdS, CodFisc, NomeS)
Materie (IdM, NomeM)
Professori (IdP, CodFisc, NomeP)
Frequenza (Classe, Studente, Anno, Esito)
Insegnamento (Classe, Materia, Anno, Professore)
Profitto (Studente, Materia, Anno, Voto)
Frequenza.Classe è in integrità referenziale con *Classi.IdC*;
Frequenza.Studente è in integrità referenziale con *Studenti.IdS*;
Insegnamento.Classe è in integrità referenziale con *Classi.IdC*;
Insegnamento.Materia è in integrità referenziale con *Materie.IdM*;
Insegnamento.Professore è in integrità referenziale con *Professori.IdP*;
Profitto.Studente è in integrità referenziale con *Studenti.IdS*;
Profitto.Materia è in integrità referenziale con *Materie.IdM*.

8 ..... **Arbitri** (Cognome, Nome, Città, Regione)
Giornate (Numero, Serie, Giorno, Mese, Anno)
Squadre (NomeSq, Città, Allenatore)
Giocatori (CodFiscale, Cognome, Nome, Ruolo, CittàNascita, Squadra)
Partite (NumeroP, NumeroG, Serie, Arbitro, Casa, Ospite, GoalCasa, GoalOspite)
Partecipazione (Giocatore, Partita, NumeroG, Serie, Ruolo)
Goal (Partita, NumeroG, Serie, Minuto, Giocatore, Autogoal)
Giocatori.Squadra è in integrità referenziale con *Squadre.Nome*Sq;
Partite.{*NumeroG*, *Serie*} è in integrità referenziale con *Giornate.*{*Numero*, *Serie*};
Partite.Arbitro è in integrità referenziale con *Arbitri.Cognome*;
Partite.Casa è in integrità referenziale con *Squadre.Nome*Sq;
Partite.Ospite è in integrità referenziale con *Squadre.Nome*Sq;
Partecipazione.Giocatore è in integrità referenziale con *Giocatori.CodFiscale*;
Partecipazione.{*Partita*, *NumeroG*, *Serie*} è in integrità referenziale con *Partite.*{*NumeroP*, *NumeroG*, *Serie*};
Goal.{*Partita*, *NumeroG*, *Serie*} è in integrità referenziale con *Partite.*{*NumeroP*, *NumeroG*, *Serie*};
Goal.Giocatore è in integrità referenziale con *Giocatori.CodFiscale*.

9 ..... **Fornitori** (IdF, Nome, Indirizzo)
Prelevatori (IdP, Nome, Indirizzo)
Arrivi (NumProgr, Data, Magazzino, Fornitore)
Prelievi (NumProgr, Data, Magazzino, Prelevatore)
Articoli (CodArticolo, Nome)
Magazzini (IdM, NomeMagazzino, Indirizzo)
Presenza (Articolo, Magazzino, Giacenza, Minimo, Massimo)
InArrivo (Arrivo, Articolo, Quantita)
InPrelievo (Prelievo, Articolo, Quantita)
Arrivi.Magazzino è in integrità referenziale con *Magazzini.IdM*;
Arrivi.Fornitore è in integrità referenziale con *Fornitori.IdF*;
Prelievi.Magazzino è in integrità referenziale con *Magazzini.IdM*;
Prelievi.Prelevatore è in integrità referenziale con *Prelevatori.IdP*;
Presenza.Articolo è in integrità referenziale con *Articoli.CodArticolo*;
Presenza.Magazzino è in integrità referenziale con *Magazzini.IdM*;
InArrivo.Arrivo è in integrità referenziale con *Arrivi.NumProgr*;
InArrivo.Articolo è in integrità referenziale con *Articoli.CodArticolo*;
InPrelievo.Prelievo è in integrità referenziale con *Prelievi.NumProgr*;
InPrelievo.Articolo è in integrità referenziale con *Articoli.CodArticolo*.

Loescher Editore - Vietata la vendita e la diffusione

Il modello fisico

6

Prerequisiti

- Organizzazione del disco
- File

Obiettivi

Conoscenze

- Componenti del DBMS per la gestione delle interrogazioni e l'accesso alla memoria secondaria
- Gestione del buffer
- Strutture primarie e secondarie
- Organizzazione sequenziale, hash e ad albero
- Progettazione fisica: azioni che si possono intraprendere a livello fisico per migliorare l'efficienza di un database relazionale
- Comportamento del gestore delle interrogazioni

Abilità/capacità

- Scegliere gli indici sulle relazioni
- Individuare vantaggi e svantaggi della presenza di un indice

Loescher Editore - Vietata la vendita e la diffusione

6.1 GESTIONE DELLE OPERAZIONI SU DISCO

Le operazioni specificate in SQL vengono affidate a un modulo detto **gestore delle interrogazioni** che traduce le interrogazioni in una forma interna, le trasforma per renderle più efficienti (**ottimizzazione** delle interrogazioni) e le realizza impiegando operazioni di livello più basso (scansione, ordinamento, accesso diretto) che fanno riferimento allo struttura fisica dei file e sono eseguite dal modulo sottostante chiamato gestore dei metodi di accesso.

Il **gestore dei metodi di accesso**, in base alla struttura fisica dei file, trasforma le richieste in operazioni di lettura e scrittura dei dati sul disco, mediate dal modulo **gestore del buffer** che mantiene temporaneamente porzioni del database in memoria centrale per maggiore efficienza.

Il gestore del buffer invia le richieste effettive di lettura e scrittura fisica al **gestore dello spazio su disco**.

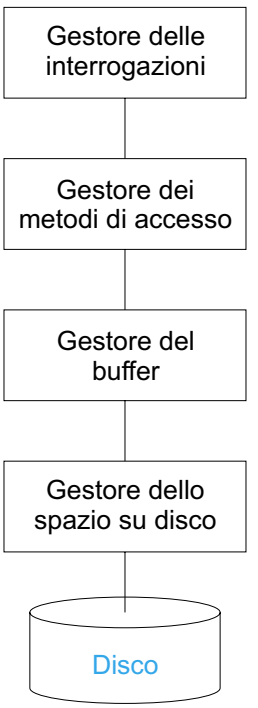


Figura 6.1
Gestione degli accessi al disco

> In pratica, per operazioni di lettura, il DBMS, dopo aver ricevuto dal programma la richiesta in cui viene specificata la chiave del record logico da elaborare, utilizzando lo schema e la descrizione fisica individua i record fisici che permettono di ricostruire il record logico desiderato.
> La richiesta di lettura dei record fisici viene trasmessa al sistema operativo che, tramite il modulo del file system, esegue l'operazione e trasferisce i dati nel buffer di I/O. Con i dati presenti nel buffer il DBMS, utilizzando lo schema, può ricostruire il record logico che viene trasferito all'area di lavoro del programma applicativo.

I **DBMS** usano le funzionalità di base del file system per creare ed eliminare file e leggere e scrivere blocchi, ma creano una propria astrazione dei file per garantire **efficienza** tramite i buffer e **transazionalità** tramite il controllore dell'affidabilità.
Tutto ciò che riguarda la **struttura dei file** è realizzato direttamente dal DBMS: un file può contenere più relazioni e le tuple di una relazione possono essere in file diversi.

Gestione del buffer

Sulla **memoria di massa** i dati sono organizzati in **blocchi**, di solito di dimensione fissa, con accesso diretto (è possibile accedere direttamente a ogni blocco ma il tempo necessario è variabile e dipende dalla posizione).

> L'interazione tra memoria centrale e memoria di massa è realizzata nei DBMS attraverso l'uso di una grande area di memoria centrale detta **buffer** gestita dal DBMS in modo condiviso per tutte le applicazioni.

Il buffer è organizzato in **pagine** che hanno dimensione pari a un numero intero di blocchi; per semplicità si può supporre che una pagina coincida con un blocco.

Il **gestore del buffer** riceve dai programmi le richieste di lettura e scrittura e si occupa di caricare e scaricare le pagine tra memoria centrale e memoria di massa.

Loescher Editore - Vietata la vendita e la diffusione

Nel caso di un'operazione di **lettura**, se la pagina è già presente nel buffer non serve una lettura fisica; se non è presente, la pagina viene letta dalla memoria di massa e caricata in una pagina libera (cioè non usata da alcun programma; per ogni pagina c'è un contatore dei programmi che la usano); se non ci sono pagine libere deve essere liberata una pagina; può essere usata la politica **steal** che libera una pagina scaricandola in memoria di massa con una operazione di flush, o la politica **no steal** che resta in attesa che si liberino pagine nel buffer.

La **scrittura** può avvenire in modo sincrono (richiamando la primitiva *force*) oppure asincrono: le informazioni scritte possono restare nel buffer, subire varie modifiche e poi essere trascritte in memoria con una sola operazione; differire la scrittura, compatibilmente con le proprietà di affidabilità del sistema, riduce il tempo di risposta.

Nota

Le politiche di gestione del buffer sono simili a quelle per la gestione della memoria centrale con paginazione dinamica.

Generalmente i buffer contengono le pagine a cui viene fatta la maggior parte degli accessi. In base al **principio di località dei dati**, i dati referenziati di recente hanno maggior probabilità di essere referenziati di nuovo nel futuro.

Principio statistico

Il 20% dei dati è tipicamente usato dall'80% delle applicazioni.

6.2 STRUTTURE PRIMARIE E SECONDARIE

Nell'**organizzazione fisica** vengono utilizzate:

- **strutture primarie** contenenti dati;
- **strutture secondarie** (**indici**): un indice secondario è un file in cui ogni record ha due campi: una **chiave** e uno o più **indirizzi** dei record che contengono quel valore nel campo indicizzato; è ordinato in base al valore della chiave e quindi facilita le ricerche per chiave (detta appunto **chiave di ricerca**), le scansioni ordinate e le ricerche di intervalli di valori.

Nota

Se la chiave di ricerca è composta da più campi (indici multicampo) l'ordinamento è effettuato in base al valore del primo campo, per valori uguali in base al secondo campo e così via.

Se l'indice contiene i dati o è realizzato su un file ordinato sullo stesso campo su cui è definito l'indice, si dice **indice primario** perché non solo garantisce l'accesso in base alla chiave ma vincola anche l'allocazione dei dati.

Su un file ci può essere solo un indice primario ma ci possono essere più indici secondari.

Le tecniche principali per l'organizzazione della **struttura primaria** di un file, che stabilisce come le tuple sono disposte nel file, si possono raggruppare in:

- **sequenziali**: collocano le tuple ordinatamente in sequenza;
- ad **accesso calcolato** (**hash**): collocano le tuple in posizioni calcolate da un algoritmo.

Le tecniche principali per la realizzazione di **indici** sono quelle ad **albero**, che però possono essere usate anche per realizzare strutture primarie.

6.3 ORGANIZZAZIONE DEI FILE

Organizzazione sequenziale

Nelle tecniche ad organizzazione sequenziale i dati sono scritti senza un ordine particolare (heap); la sequenza delle tuple dipende dall'ordine di immissione.

Loescher Editore - Vietata la vendita e la diffusione

La scrittura di nuovi dati avviene sempre alla fine del file, quindi le operazioni di **inserimento** sono molto efficienti.
Le operazioni di **cancellazione** però lasciano spazio inutilizzato all'interno del file; le operazioni di **modifica** dei dati possono creare grossi problemi se cambia la dimensione del record da variare.

L'**organizzazione sequenziale** è ottimale per svolgere letture e scritture sequenziali ma risulta inefficiente per effettuare ricerche perché richiede sempre la scansione di tutto il file.

L'organizzazione sequenziale è molto usata perché è facile da gestire; per rendere più efficiente la ricerca vengono poi aggiunti **indici secondari**.

Tipi particolari di organizzazioni sequenziali sono quelle **ad array**, in cui ogni tupla è identificata da un valore che fa da indice, e quelle **ordinate** in cui le tuple sono disposte in ordine di chiave.

Accesso calcolato

Nelle tecniche ad **accesso calcolato** (**hash**) l'indirizzo di ciascun elemento all'interno dell'area di memoria riservata viene calcolato applicando una funzione (**funzione indirizzo**) alla chiave che identifica l'elemento.

La stessa funzione viene utilizzata per reperire l'elemento in fase di ricerca.

La funzione indirizzo non garantisce che gli indirizzi associati agli elementi siano tutti diversi tra loro: a due elementi con chiavi diverse può venire associato lo stesso indirizzo; tali elementi vengono detti **sinonimi** (si parla anche di **collisioni**).

Ci sono diversi modi per gestire le collisioni.
Con il metodo a indirizzamento aperto i sinonimi vengono allocati nella prima posizione successiva libera; per cercare un record si applica la funzione alla chiave di ricerca ma, se l'indirizzo restituito dalla funzione indirizzo è occupato da un altro elemento (con chiave diversa), la ricerca deve proseguire in tutte le posizioni successive fino alla prima posizione vuota; se è permessa la cancellazione, trovare una posizione libera non garantisce il termine della ricerca.
Una soluzione migliore consiste nell'utilizzo di **catene di overflow**: i record sono raggruppati in bucket; ogni bucket comprende tutti gli elementi che sono tra loro sinonimi ed è formato da una pagina primaria e da pagine aggiuntive (pagine di overflow) allocate quando necessario, collegate in catena; per cercare un record si applica la funzione alla chiave di ricerca per identificare il bucket e poi basta scandire i record della pagina primaria e delle eventuali pagine di overflow collegate.

Una funzione che associa a un dato un valore numerico è detta **funzione di hash**.

La **funzione indirizzo** è una funzione di hash che interpreta il numero restituito come un indirizzo di memoria.
Perché una funzione indirizzo sia buona, deve permettere di ottenere una buona utilizzazione della memoria e deve generare pochi sinonimi.
I valori prodotti devono appartenere (o essere ricondotti) ad un intervallo desiderato, che individua la zona di memoria in cui i dati vengono registrati.

Le funzioni indirizzo in genere operano sulle chiavi producendo dei valori pseudo-casuali, tramite operazioni matematiche se la chiave è numerica.
Se la chiave è alfanumerica si può ricorrere a qualche artificio che produca valori numerici dalla sequenza dei caratteri (per esempio sostituendo ad ogni carattere il suo codice ASCII).

L'**organizzazione ad accesso calcolato** è ideale per accessi puntuali tramite la chiave ma risulta inefficiente per ricerche di intervalli di valori perché le funzioni di hash tendono a sparpagliare valori vicini su blocchi diversi.

Loescher Editore - Vietata la vendita e la diffusione

L'organizzazione ad accesso calcolato non dà nessun beneficio per ricerche su campi diversi dal campo chiave.

Nota

Strutture ad albero

Le strutture ad **albero** possono essere utilizzate per strutture primarie (dati o indici primari), ma sono utili soprattutto per gli **indici secondari** perché possono essere aggiunte o eliminate dinamicamente ai file contenenti i dati.

Le **strutture ad albero** rendono più efficienti sia accessi puntuali che per intervalli di valori.

Le strutture ad albero più usate sono quelle di tipo **B** (B-tree) e **B+** (B+-tree).
Sono strutture dinamiche, cioè efficienti anche in presenza di frequenti aggiornamenti.
Gli alberi B+ sono i più usati, ma anche gli alberi B garantiscono una efficienza elevata.

B-tree

In un **albero B** (**B-tree**) ogni nodo corrisponde a una pagina o blocco; i legami tra i nodi sono costituiti da puntatori che collegano fra loro le pagine.
Ogni nodo ha molti figli (anche centinaia); questo consente di avere un numero limitato di livelli, con la maggioranza delle pagine occupate da nodi foglia.

Se il numero di livelli è ridotto la ricerca è più veloce.

Nota

Gli alberi sono **bilanciati** cioè la lunghezza di un cammino dalla radice a un qualsiasi nodo foglia è costante.
Il tempo di accesso alle informazioni è lo stesso per tutte le foglie e dipende dalla profondità dell'albero.

Ogni **nodo non foglia** contiene una sequenza di *n* valori della chiave ordinati.
Ogni chiave K_i con *i* tra 1 e *n* è seguita da un puntatore P_i; K_1 è preceduta da un puntatore P_0.

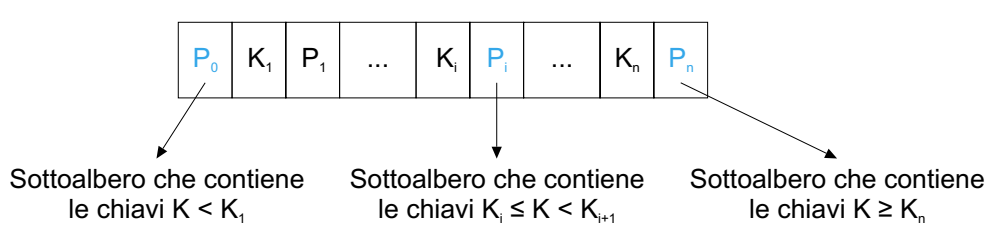


Figura 6.2 *Struttura di un nodo non foglia*

Ciascun puntatore indirizza un sottoalbero con le seguenti caratteristiche:

- P_0 indirizza un sottoalbero che contiene l'informazione relativa alle chiavi strettamente inferiori a K_1;
- P_n indirizza un sottoalbero che contiene l'informazione relativa alle chiavi superiori o uguali a K_n;
- ogni P_i indirizza un sottoalbero che contiene l'informazione relativa alle chiavi superiori o uguali a K_i e strettamente inferiori a K_{i+1} (tutti i valori di K tali che $K_i \leq K < K_{i+1}$).

Perciò ogni nodo contiene *n* valori di chiave e *n+1* puntatori.
Il valore *n+1* è detto **fan-out** dell'albero.
Il valore *n* dipende dall'ampiezza della pagina e dalla dimensione delle chiavi e dei puntatori.
Si sceglie per *n* il valore massimo possibile in modo da ridurre il numero di livelli dell'albero, però i vari nodi possono contenere meno di *n* valori per flessibilità.

Loescher Editore - Vietata la vendita e la diffusione

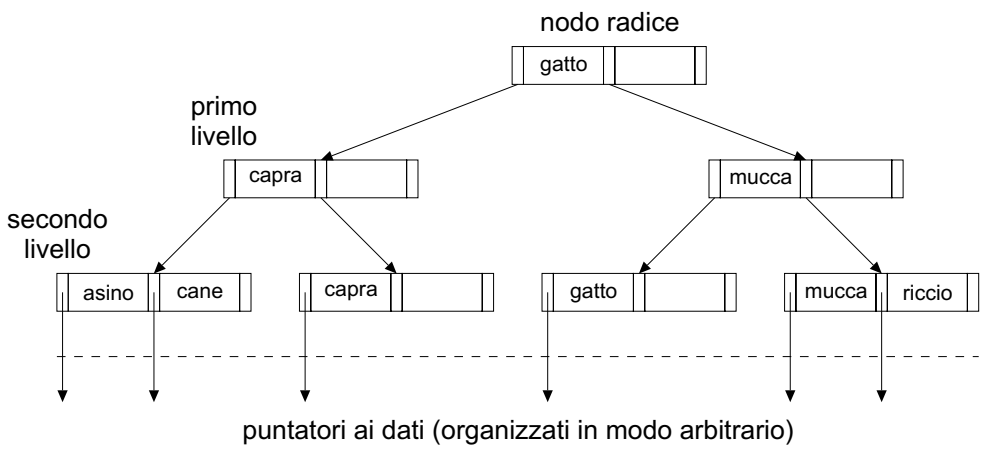


Figura 6.3 *Esempio di B-tree*

La **ricerca di una chiave** V segue i puntatori partendo dalla radice; a ogni nodo intermedio:

- se $V < K_1$ si segue P_0;
- se $V >= K_n$ si segue P_n;
- altrimenti si segue P_i tale che $K_i <= V < K_{i+1}$.

I **nodi foglia** possono:

- **contenere la tupla** allora la struttura è detta **key-sequenced** e il file è ordinato; la posizione della tupla nel nodo dipende dal valore della chiave; questo tipo di struttura è usata solo per indici primari;
- contenere **puntatori** ai blocchi che contengono le tuple; la struttura è detta indiretta e la posizione delle tuple nei blocchi può essere qualsiasi; questo tipo di struttura si presta per indici secondari.

Operazioni di **aggiornamento** di una relazione, con inserimenti, modifiche e cancellazioni di tuple, provocano anche aggiornamenti degli indici correlati.

L'aggiornamento di un albero a causa dell'**inserimento** di una tupla è facile se la chiave va in una foglia in cui c'è spazio, altrimenti si deve ricorrere ad una operazione di split (divisione) che crea due foglie; tale operazione richiede la modifica dei puntatori e altri eventuali split anche ai livelli superiori.

La **cancellazione** di una tupla può essere fatta marcando la tupla come non valida; in tal caso gli indici non vengono modificati.
Se le tuple vengono cancellate effettivamente, i valori delle chiavi relative devono essere cancellati dai nodi dell'albero. Se vengono cancellati molti valori e le pagine risultano poco utilizzate si può fare il merge (fusione) di più pagine.

La **modifica della chiave** di una tupla viene trattata come una cancellazione della tupla più un inserimento della tupla con la nuova chiave.

A volte, a causa di operazioni di aggiornamento, può essere necessario **ribilanciare** l'albero.

Spesso le pagine dei primi livelli dell'albero risiedono nel **buffer** per aumentare l'efficienza.
Le pagine dei primi livelli possono diventare risorse delicate ai fini della concorrenza se sono soggette a molti aggiornamenti, ma di solito vengono solo lette.

B+-TREE

Negli **alberi B+** (**B+-tree**) i nodi foglia sono collegati tra di loro da una catena in base all'**ordine della chiave**, facilitando scansioni ordinate e selezioni per intervallo.

Loescher Editore - Vietata la vendita e la diffusione

Gli alberi B+ possono essere usati sia se i nodi foglia contengono le tuple, che se contengono solo puntatori ai blocchi che contengono le tuple, cioè sia per strutture key-sequenced che per strutture indirette.

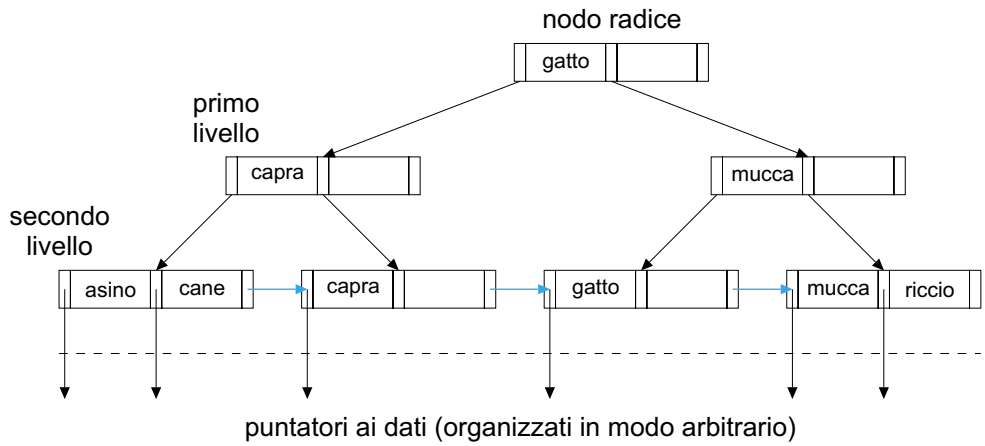


Figura 6.4 *Esempio di B+-tree*

6.4 PROGETTAZIONE FISICA

La **progettazione fisica** avviene in base allo schema logico e alle previsioni del carico applicativo.

La progettazione fisica può essere molto complessa; prevede, oltre alle scelte delle strutture fisiche, anche molti parametri come dimensioni iniziali, possibilità di espansione, contiguità di allocazione, quantità e dimensioni dei buffer e così via.
Le scelte dipendono dallo specifico sistema di gestione utilizzato, che stabilisce i parametri e fornisce valori standard.

SCELTE FONDAMENTALI DI PROGETTAZIONE FISICA

- scelta della struttura primaria per ciascuna relazione tra quelle disponibili dal DBMS;
- definizione di indici secondari.

Le operazioni da considerare per decidere le strutture fisiche da utilizzare sono **selezioni** e **join** che diventano più efficienti se sui campi interessati è definito un indice primario o secondario o una struttura hash.

Spesso nelle operazioni sul database sono coinvolte chiavi, quindi è ragionevole definire su ogni relazione un **indice** o una struttura **hash** in corrispondenza della **chiave**.

Nota I DBMS di solito costruiscono automaticamente un indice sulla chiave primaria.

Può essere utile definire **indici** su altri campi su cui vengono effettuate selezioni, oppure su cui è richiesto un ordinamento.

Nota Bisogna ricordare però che le strutture hash e gli indici primari possono avere comportamenti non desiderabili in caso di aggiornamenti frequenti.

La scelta degli indici si basa spesso su un procedimento empirico per prove successive: dopo la sperimentazione del funzionamento si vede se le prestazioni sono buone, altrimenti si possono aggiungere altri indici o eliminarne alcuni.

Loescher Editore - Vietata la vendita e la diffusione

6.5 IL GESTORE DELLE INTERROGAZIONI

Il **gestore delle interrogazioni** è il modulo responsabile dell'esecuzione efficiente delle operazioni.

Ogni **interrogazione SQL** viene analizzata in base al dizionario dei dati e, se accettata, viene tradotta in una **forma interna** di tipo algebrico; dopo una fase di **ottimizzazione** viene generato il codice che usa i metodi di accesso ai dati.

Per effettuare l'**ottimizzazione**, il gestore delle interrogazioni deve scegliere quali operazioni di accesso ai dati svolgere (per esempio tra una scansione e un accesso tramite indici), l'ordine delle operazioni (per esempio tra più join presenti in una interrogazione), quale alternativa usare per realizzare una operazione (per esempio quale metodo di join), a quale livello della strategia eseguire un eventuale ordinamento, e così via.

Il gestore delle interrogazioni costruisce un **albero delle alternative**; ogni nodo foglia corrisponde a una strategia di esecuzione, descritta dalle scelte del cammino. L'alternativa migliore è individuata dal cammino alla foglia con costo minore.

In realtà non si cerca la soluzione ottima, ma ci si accontenta di soluzioni buone (dello stesso ordine di grandezza prestazionale della soluzione ottima), per ridurre i tempi del lavoro di ottimizzazione.

L'**ottimizzazione** agisce a tempo di compilazione.
Possono essere usate in alternativa due tecniche:

- **compile and store**: il codice prodotto viene memorizzato nel database; se il database varia in modo significativo, per esempio per l'aggiunta di un indice, la compilazione dell'interrogazione viene invalidata e ripetuta;
- **compile and go**: il codice prodotto viene eseguito subito, senza memorizzazione.

Per valutare l'efficienza delle operazioni viene effettuata una **analisi statistica** che definisce l'ordine di grandezza delle **dimensioni dei risultati intermedi**, per esempio il numero di pagine occupate.

Il **join** è l'operazione più gravosa.

Esistono vari algoritmi per l'esecuzione del join; le strategie principali sono:

- **nested loop**: una tabella viene definita come esterna e una come interna; si inizia una scansione sulla tabella esterna e per ogni tupla si cercano le tuple corrispondenti della tabella interna (cioè le tuple con lo stesso valore negli attributi di join); in questo caso è opportuno un indice sulla tabella interna sull'attributo di join, altrimenti servirebbe una scansione per ogni valore di join della tabella esterna (nested loop significa infatti scansione nidificata); il costo varia in base a quale tabella è scelta come esterna o interna;
- **merge scan**: entrambe le tabelle devono essere ordinate in base ai valori degli attributi di join; vengono eseguite in parallelo due scansioni sulle tuple delle due tabelle; è una strategia efficiente per tabelle ordinate o con indici; per applicare il merge scan può essere necessario prima ordinare le tabelle (**sort-merge scan**) ma se il numero di tuple è piccolo può essere comunque conveniente.

Nota

Il costo di ciascuna strategia non può essere valutato in astratto, ma dipende dalla situazione iniziale, come la presenza di indici, e dalla situazione finale, per esempio il risultato ordinato.

Loescher Editore - Vietata la vendita e la diffusione

VERIFICA

Questionario

1. Che cos'è il gestore delle interrogazioni?
2. Di che cosa si occupa il gestore dei metodi di accesso?
3. Che cos'è il buffer?
4. Di che cosa si occupa il gestore del buffer?
5. Che cosa dice il principio di località dei dati?
6. Che cosa sono gli indici?
7. Che cos'è un indice primario?
8. Come funziona l'organizzazione sequenziale?
9. Come funziona l'organizzazione ad acceso calcolato?
10. Come funziona l'organizzazione ad albero?
11. In che cosa consiste la progettazione fisica?
12. In che cosa consiste l'ottimizzazione delle interrogazioni?
13. Quali sono gli algoritmi principali per l'esecuzione di una operazione di join?

Test

1 ..... *Indicare se le seguenti affermazioni sono vere o false.*

V	F	
☐	☐	1) Un file di dati è una struttura primaria.
☐	☐	2) Le strutture primarie determinano l'ordine dei dati nel file.
☐	☐	3) Gli indici permettono di velocizzare tutte le operazioni sui file.
☐	☐	4) Un indice primario contiene i dati del file.
☐	☐	5) Un indice secondario contiene i dati del file.
☐	☐	6) Su un file ci possono essere più indici primari.
☐	☐	7) Su un file ci possono essere più indici secondari.
☐	☐	8) Nell'organizzazione sequenziale la posizione dei dati dipende dall'ordine di immissione.
☐	☐	9) Nell'organizzazione ad accesso calcolato la posizione dei dati dipende dall'ordine di immissione.
☐	☐	10) Una funzione indirizzo associa un numero a ogni dato.
☐	☐	11) Una funzione indirizzo può restituire lo stesso valore per dati diversi.
☐	☐	12) Le strutture ad albero sono usate soprattutto per gli indici secondari.
☐	☐	13) Un B-tree può avere rami di lunghezza molto diversa tra loro.
☐	☐	14) Il fan-out di un albero è il numero di puntatori contenuti in un nodo dell'albero.
☐	☐	15) L'aggiornamento dei dati non causa la modifica degli indici associati al file.
☐	☐	16) Un B-tree è un albero binario.
☐	☐	17) In un albero B+ i nodi foglia sono collegati tra di loro.

Loescher Editore - Vietata la vendita e la diffusione

V	F	
☐	☐	18) In un albero B+ tutti i nodi dello stesso livello sono collegati tra di loro.
☐	☐	19) Gli alberi B e B+ non possono contenere i dati.
☐	☐	20) Di solito si crea un indice sulla chiave della relazione.

2 ..... *Segnare tutte le affermazioni esatte.*

1) L'organizzazione sequenziale:
- ☐ facilita l'inserimento dei dati;
- ☐ facilita le ricerche del valore di una chiave;
- ☐ facilita le scansioni ordinate e le ricerche per intervalli di valori;
- ☐ può essere usata per strutture primarie;
- ☐ può essere usata per strutture secondarie.

2) L'organizzazione ad accesso calcolato:
- ☐ facilita l'inserimento dei dati;
- ☐ facilita le ricerche del valore di una chiave;
- ☐ facilita le scansioni ordinate e le ricerche per intervalli di valori;
- ☐ può essere usata per strutture primarie;
- ☐ può essere usata per strutture secondarie.

3) L'organizzazione ad albero:
- ☐ facilita l'inserimento dei dati;
- ☐ facilita le ricerche del valore di una chiave;
- ☐ facilita le scansioni ordinate e le ricerche per intervalli di valori;
- ☐ può essere usata per strutture primarie;
- ☐ può essere usata per strutture secondarie.

PROPOSTE DI LAVORO

1 ..... Scrivere un programma che realizzi una funzione di hash che riceva come parametro un numero e che restituisca sempre un valore minore di 100.
(Suggerimento: basta calcolare il resto della divisione del numero per 100.)

2 ..... Scrivere un programma che realizzi una funzione di hash che riceva come parametro una stringa e che restituisca sempre un valore minore di 100.
(Suggerimento: sommare i numeri corrispondenti ai caratteri della stringa, 1 per a, 2 per b ecc. e poi calcolare il resto della divisione della somma per 100.)

3 ..... Scrivere un programma che simuli l'allocazione ad accesso calcolato di stringhe in una matrice di 100 righe, ognuna di 10 elementi, per permettere l'allocazione di 10 sinonimi. Effettuare la ricerca di una stringa nella struttura così creata.

4 ..... Disegnare una struttura ad albero bilanciato che contenga i nomi di alcuni animali. Introdurre un dato che provochi lo split di un nodo al livello foglia e descrivere cosa accade al livello foglia e al livello superiore.

5 ..... Per le relazioni create negli esempi e nelle proposte di lavoro dei capitoli precedenti individuare gli indici più utili.

Loescher Editore - Vietata la vendita e la diffusione

Modulo 3

DBMS

7 MySQL

8 Access

Loescher Editore - Vietata la vendita e la diffusione

7 MySQL

Prerequisiti

- DBMS relazionali
- Linguaggio SQL

Obiettivi

Conoscenze

- Caratteristiche di MySQL
- Clausole specifiche di MySQL
- Trigger, stored routine, controllo dell'accesso e transazioni in MySQL
- Strumenti di gestione e amministrazione a interfaccia grafica

Abilità/capacità

- Installare e avviare il server MySQL
- Usare MySQL per la definizione e l'inserimento di dati, la loro modifica e l'esecuzione di interrogazioni sul database
- Saper utilizzare i costrutti avanzati di MySQL: integrare lo schema con procedure e trigger, attribuire le autorizzazioni agli utenti, controllare l'esecuzione delle transazioni
- Utilizzare gli strumenti a interfaccia grafica MySQL Query Browser, MySQL Administrator e phpMyAdmin

Loescher Editore - Vietata la vendita e la diffusione

7.1 IL DBMS MYSQL

MySQL è un DBMS relazionale sviluppato dalla società svedese MySQL AB, recentemente acquisita da Sun Microsystem.
MySQL nella versione **Community Server** è un software **open source** che può essere utilizzato gratuitamente secondo la licenza **GNU GPL** (GNU General Public License) che assicura il diritto di usare, modificare e distribuire il software, purché si ridistribuisca il software prodotto con gli stessi diritti (bisogna quindi rilasciare il codice sorgente della propria applicazione se si intende distribuirla).

La versione Enterprise, che integra diversi tool di amministrazione e diagnostica, è orientata alle aziende e richiede l'acquisto di una licenza commerciale, ma può essere utilizzata per la realizzazione di software proprietari. **Nota**

MySQL è molto utilizzato in particolare in ambito Web, in combinazione con server Web Apache e applicazioni PHP, sia in sistemi Linux che Windows. Per descrivere queste piattaforme sono stati coniati gli acronimi **LAMP** (Linux Apache MySQL PHP, ma anche Python o PERL) e **WAMP** (Windows Apache MySQL PHP).
MySQL può essere usato anche in applicazioni realizzate con vari linguaggi di programmazione, tra cui Java e C/C++, grazie anche alla disponibilità di un'interfaccia ODBC (**MyODBC**).

VERSIONI DI MYSQL

MySQL nasce nel 1995, ma è reso disponibile solo a partire dal 2000 con le prime beta della versione 3.23.
Tra il 2002 e il 2003 viene rilasciata la versione **4.0** che introduce tra l'altro il supporto alle query UNION.
Nel 2003 esce la versione **4.1**, che offre molte nuove funzionalità, la più importante delle quali è probabilmente il supporto alle query SELECT nidificate.
La versione **5.0**, disponibile sin dal 2003, ma considerata stabile a partire dal 2005, introduce significative novità, quali viste, procedure e trigger.

Nel seguito del testo si farà riferimento alla versione 5.0 di MySQL. **Nota**

È stata rilasciata anche la versione 5.1, che introduce partizionamento delle tabelle e schedulazione di eventi direttamente tramite il DBMS. **Nota**

ARCHITETTURA CLIENT/SERVER

MySQL ha un'architettura di tipo **client/server**.
Il **client** consente di inviare interrogazioni al server e si occupa di presentare all'utente i dati restituiti dal server.
Il **server** risponde alle interrogazioni dell'utente e si occupa della gestione e della memorizzazione dei dati, rendendo queste operazioni trasparenti.

Il programma client da linea di comando si chiama ***mysql***, il server si chiama ***mysqld*** (MySQL Daemon).

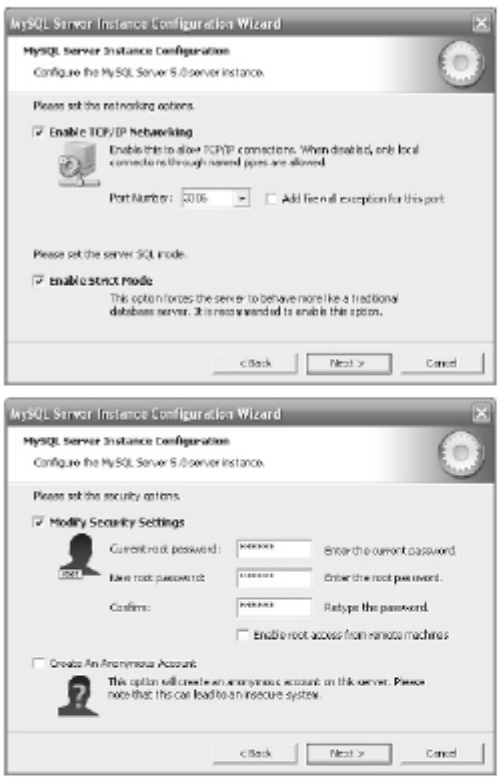
Oltre ai programmi da linea di comando sono disponibili sia come client che per la gestione del server strumenti a interfaccia grafica come *MySQL Query Browser*, *MySQL Administrator* e *phpMyAdmin*.

Loescher Editore - Vietata la vendita e la diffusione

INSTALLAZIONE

In ***Linux*** MySQL di solito viene già installato insieme al sistema operativo; la configurazione del server viene effettuata dopo l'installazione tramite lo strumento a riga di comando *mysqld*.
In ***Windows*** MySQL è installabile facilmente tramite il tool grafico *MySQL Server Setup Wizard*. *MySQL Server Instance Config Wizard* permette di modificare le impostazioni di configurazione del server, scegliendo in particolare la porta su cui il server è in ascolto, le impostazioni di avvio e una password per l'utente root.

Figura 7.1 *Alcune fasi di configurazione del server MySQL*



AVVIO

Il **server MySQL** viene eseguito come **servizio** (processo in background in attesa di richieste).

In *Linux* MySQL può essere avviato mediante lo script ***mysqld*** con l'argomento *start*, eseguito da root.

```
/etc/rc.d/init.d/mysqld start
```

La directory che contiene gli script per l'attivazione dei servizi può essere diversa in base alla distribuzione (*/etc/rc.d/* o */etc/init.d/*) **Nota**

In *Windows* si può gestire il server MySQL dall'icona *Servizi* degli *Strumenti di amministrazione* nel *Pannello di controllo*; si può impostare l'avvio manuale o automatico e avviarlo o arrestarlo.

Figura 7.2
Gestione del servizio MySQL in Windows

Per l'amministrazione del server si può usare il comando *mysqladmin* ma è più semplice utilizzare strumenti a interfaccia grafica. **Nota**

Loescher Editore - Vietata la vendita e la diffusione

Client

Per eseguire il client mysql da linea di comando basta richiamare il comando ***mysql***, in *Linux* da una finestra di terminale, in *Windows* dalla finestra di prompt dei comandi (accessibile da *Start*, *Esegui...*, digitando *cmd*, o disponibile anche ta gli *Accessori*).

Nota

Per richiamare *mysql* senza specificarne il percorso, la sottodirectory *bin* della directory di installazione di MySQL deve essere inclusa nella variabile d'ambiente PATH.
In *Windows* durante l'installazione viene creata una voce anche nel menu programmi (*Start*, *Programmi*, *MySQL*, *MySQL Server 5.0*, *MySQL Command Line Client*).

Le opzioni principali del comando ***mysql*** sono:

-h *host*	indica un socket (combinazione *hostname:porta* o *indirizzoIp:porta*) al quale connettersi; se non specificato, per default si intende localhost:3306;
-u *user*	indica il nome utente con cui effettuare il login al server MySQL; se non specificato viene utilizzato il nome usato dall'utente per il login al sistema;
-p	presenta un prompt per l'inserimento della password dell'utente; se non specificato, si intende che l'utente vuole accedere senza specificare alcuna password.

Una volta effettuato l'accesso viene presentato un prompt per l'inserimento di comandi in modo interattivo.

Accesso a MySQL sul computer locale come utente root, senza password.

```
>mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.0.67-community

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Per chiudere la connessione e uscire dal programma si possono usare i comandi **QUIT** o **EXIT**.

Il comando **HELP** visualizza l'elenco dei comandi disponibili.
Si possono ottenere informazioni più dettagliate su un comando digitando *HELP nomeComando*.

```
mysql> HELP

For the complete MySQL Manual online, visit:
   http://www.mysql.com/documentation

For info on technical support from MySQL developers, visit:
   http://www.mysql.com/support

For info on MySQL books, utilities, consultants, etc., visit:
   http://www.mysql.com/portal

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?         (\?) Synonym for 'help'.
clear     (\c) Clear command.
connect   (\r) Reconnect to the server. Optional arguments are db
and host.
delimiter (\d) Set query delimiter.
ego       (\G) Send command to mysql server, display result
```

Loescher Editore - Vietata la vendita e la diffusione

```
vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
notee     (\t) Don't write into outfile.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source    (\.) Execute a SQL script file. Takes a file name as an
argument.
status    (\s) Get status information from the server.
tee       (\T) Set outfile [to_outfile]. Append everything into
given outfile.
use       (\u) Use another database. Takes database name as
argument.

For server side help, type 'help contents'
```

Il comando **HELP CONTENTS** presenta una lista di categorie dei comandi che agiscono sul server. Si può accedere alla guida di una categoria digitando *HELP nomeCategoria*.

```
mysql> HELP CONTENTS
You asked for help about help category: "Contents"
For more information, type 'help <item>', where <item> is one of
the following
categories:
   Account Management
   Administration
   Data Definition
   Data Manipulation
   Data Types
   Functions
   Functions and Modifiers for Use with GROUP BY
   Geographic Features
   Language Structure
   Plugins
   Storage Engines
   Stored Routines
   Table Maintenance
   Transactions
   Triggers
```

I **comandi** possono essere scritti sia in minuscolo che in maiuscolo.
I comandi principali (quelli restituiti da HELP) non richiedono caratteri di terminazione; tutti gli altri comandi, per esempio le query, devono terminare con il **punto e virgola** (;); se non si scrive il punto e virgola alla pressione del tasto invio viene presentato il **prompt di continuazione** (->); quando si scrive il punto e virgola il comando viene terminato e inviato.

Nota

In questo modo si possono per esempio scrivere query su più righe per migliorarne la leggibilità.

Loescher Editore - Vietata la vendita e la diffusione

Se il comando è scritto in modo errato si ottiene un messaggio di errore e si deve riscrivere il comando. Il tasto ↑ permette di ottenere man mano le righe digitate in precedenza, è possibile selezionare la riga desiderata e accettarla o modificarla.

I comandi di uso più comune sono:

SHOW DATABASES	elenca i nomi dei database accessibili dall'utente;
CREATE DATABASE nomeDatabase	crea un nuovo database con il nome specificato;
DROP DATABASE nomeDatabase	cancella un database con tutti i dati contenuti; la cancellazione è irreversibile;
USE nomeDatabase	permette di scegliere un database;
SHOW TABLES	elenca i nomi delle tabelle del database scelto
DESCRIBE nomeTabella	visualizza la struttura della tabella specificata.
TEE nomeFile	permette di creare un file di log di tutto ciò che appare nella finestra dei comandi, finché non si digita *\t* (il file creato è un file di testo);
SOURCE nomeFile	permette di specificare un file che contiene comandi MySQL per mandarli in esecuzione in modo batch.

Nota

Alcuni comandi richiedono i permessi di amministratore.

Esempio 7.1

Esempio di una sessione che crea un database, crea una tabella nel database, inserisce alcuni dati e visualizza le informazioni relative.

```
[root@PC root]# mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.0.67-community

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> CREATE DATABASE dati;
Query OK, 1 row affected (0.04 sec)

mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| dati     |
+----------+
4 rows in set (0.00 sec)

mysql> USE dati;
Database changed

mysql> CREATE TABLE elenco (
    -> citta varchar(20),
    -> cap varchar(5)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

Loescher Editore - Vietata la vendita e la diffusione

```
mysql> SHOW TABLES;
+----------------+
| Tables_in_dati |
+----------------+
| elenco         |
+----------------+
1 row in set (0.00 sec)

mysql> DESCRIBE elenco;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| citta | varchar(20) | YES  |     | NULL    |       |
| cap   | varchar(5)  | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> INSERT INTO elenco (citta, cap)
    -> values ('Bologna', '40100');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * from elenco;
+---------+-------+
| citta   | cap   |
+---------+-------+
| Bologna | 40100 |
+---------+-------+
1 row in set (0.00 sec)
```

7.2 PRIMI ELEMENTI

Gli identificatori sono case sensitive.

Tipi di dato in MySQL

Tipi numerici.

Tutti i tipi numerici supportano l'attributo UNSIGNED per usare solo numeri senza segno.

Tipi numerici interi:

Tipo	Valori SIGNED	Valori UNSIGNED
BOOLEAN	0,1	0,1
TINYINT	-128 .. 127	0 .. 255
SMALLINT	-32768 .. 32767	0 .. 65535
MEDIUMINT	-8388608 .. 8388607	0 .. 16777215
INT o INTEGER	-2147483648 .. 2147483647	0 .. 4294967295
BIGINT	-9E+18 .. 9E+18	0 .. 18E+18

Nota

BOOLEAN ha due valori, true e false. In realtà questi valori sono sinonimi di 0 e 1, quindi questo tipo è considerato un tipo numerico.

Loescher Editore - Vietata la vendita e la diffusione

Esiste anche il tipo SERIAL, sinonimo di BIGINT UNSIGNED AUTO_INCREMENT. **Nota**

Tipi numerici decimali:

FLOAT	numero floating point con valori da -3.402823466E+38 a -1.175494351E-38 e da 1.175494351E-38 a 3.402823466E+38;
DOUBLE	numero floating point con valori da -1.7976931348623157E+308 a -2.2250738585072014E-308 e da 2.2250738585072014E-308 a 1.7976931348623157E+308;
DECIMAL(M,D)	numero a virgola fissa con *M* cifre di cui *D* decimali, per default M=10 e D=0.

Se è specificato UNSIGNED, i valori disponibili sono solo quelli positivi. **Nota**

Tipi data/ora

TIME	orario nel formato *HH:MM:SS*; i valori rappresentabili sono compresi tra -838:59:59 e 838:59:59;
DATE	data nel formato *YYYY-MM-DD*;
DATETIME	data nel formato di DATE e orario nel formato di TIME; i valori dell'orario però sono limitati tra 00:00:00 e 23:59:59;
TIMESTAMP	numero intero non negativo che rappresenta il numero di secondi trascorsi dalla mezzanotte del primo gennaio 1970;
YEAR(2) YEAR(4)	anno nella notazione a 2 o 4 cifre.

Tipi carattere e stringa

CHAR(M)	campo di caratteri non case sensitive di dimensione fissa *M*, con *M* da 0 a 255, default 1; se la stringa inserita è più lunga viene troncata, se è più corta, viene completata con spazi;
BINARY(M)	equivalente a CHAR(M) ma case sensitive;
VARCHAR(M)	campo di caratteri non case sensitive di dimensione variabile *M*, con *M* da 0 a 255, default 1; se la stringa inserita è più lunga viene troncata;
VARBINARY(M)	equivalente a VARCHAR(M) ma case sensitive;
TEXT	equivalente a VARCHAR(65535);
BLOB	equivalente a VARBINARY(65535);
ENUM('v1','v2',...,'vn')	indica che il campo può assumere esattamente un valore tra quelli elencati (al massimo 65535);
SET('v1','v2',...,'vn')	indica che il campo può assumere un insieme di valori (0 o più) tra quelli elencati (al massimo 65535).

Esistono anche i tipi TINYTEXT, TINYBLOB, MEDIUMTEXT, MEDIUMBLOB, LONGTEXT e LONGBLOB che contengono fino a 255, 16.777.215 e 4.294.967.295 di caratteri. **Nota**

I campi di testo TEXT, BLOB, TINYTEXT, MEDIUMTEXT e LONGTEXT non possono essere indicizzati, di conseguenza non possono far parte di chiavi primarie o esterne. **Nota**

Il tipo ENUM

Se si inserisce un valore non valido in un campo di tipo ENUM viene inserita la stringa vuota. Se il campo permette valori NULL il valore di default è NULL, altrimenti è il primo elemento della lista.

Loescher Editore - Vietata la vendita e la diffusione

Operatori e funzioni

- Operatori aritmetici: + - * / DIV (quoziente nella divisione tra interi) % (modulo, resto della divisione tra interi)

```
mysql> SELECT 15/2, 15 div 2, 15%2;
+--------+----------+------+
| 15/2   | 15 div 2 | 15%2 |
+--------+----------+------+
| 7.5000 |        7 |    1 |
+--------+----------+------+
```

- operatori logici: AND, OR e NOT (utilizzabili anche con la sintassi C: &&, || e !)
- operatori di confronto: = <> o != < > <= >=

Sono supportati anche gli operatori IS NULL, BETWEEN, IN e LIKE.

Nota

MySQL rende disponibile anche l'operatore REGEXP che cerca la presenza di sottostringhe che rispettano il modello descritto da un'espressione regolare.

Sono disponibili le funzioni di aggregazione COUNT(), SUM(), AVG(), MIN() e MAX().

Funzioni matematiche

ABS(x)	valore assoluto;
CEIL(x) o CEILING(x)	numero intero più piccolo maggiore o uguale a *x*;
EXP(x)	e^x;
FLOOR(x)	numero intero più grande minore o uguale a *x*;
LN(x)	logaritmo naturale (in base *e*);
LOG10(x)	logaritmo in base 10;
LOG2(x)	logaritmo in base 2;
PI()	valore di pi greco;
POW(x,y) o POWER(x,y)	potenza di base *x* ed esponente *y*;
RAND()	restituisce un valore casuale compreso tra 0 e 1;
RAND(n)	restituisce un valore casuale compreso tra 0 e 1, con seme *n*;
ROUND(x)	numero intero più prossimo a x;
SQRT(X)	radice quadrata;
TRUNCATE(x,d)	*x* troncato a *d* cifre decimali.

```
mysql> SELECT CEIL(9.9), FLOOR(9.9), ROUND(9.9), TRUNCATE(9.9,0);
+-----------+-----------+-----------+-----------------+
| CEIL(9.9) | FLOOR(9.9) | ROUND(9.9) | TRUNCATE(9.9,0) |
+-----------+-----------+-----------+-----------------+
|        10 |         9 |        10 |               9 |
+-----------+-----------+-----------+-----------------+

mysql> SELECT ABS(-3), LOG2(16), PI(), POW(5,3), SQRT(144);
+---------+----------+----------+----------+-----------+
| ABS(-3) | LOG2(16) | PI()     | POW(5,3) | SQRT(144) |
+---------+----------+----------+----------+-----------+
|       3 |        4 | 3.141593 |      125 |        12 |
+---------+----------+----------+----------+-----------+
```

Loescher Editore - Vietata la vendita e la diffusione

Funzioni per l'elaborazione delle date

CURDATE()	restituisce la data corrente nel formato *YYYY-MM-DD*;
CURTIME()	restituisce l'orario corrente nel formato *HH:MM:SS*;
DATEDIFF(d1, d2)	numero di giorni che intercorre tra due date (*d1* - *d2*);
DATE_ADD(d, i) e DATE_SUB(d, i)	rispettivamente aggiungono o tolgono alla data *d* l'intervallo *i*;
DATE_FORMAT(d, f)	converte una data in una stringa in base al formato *f*;
DAY(d) o DAYOFMONTH(d)	giorno del mese (0 .. 31);
DAYNAME(d)	giorno della settimana;
DAYOFYEAR(d)	giorno dell'anno (1 .. 366);
HOUR(t)	ora;
MINUTE(t)	minuti;
MONTH(d)	mese (1 .. 12);
MONTHNAME(d)	nome del mese;
NOW()	data e ora di inizio dell'istruzione;
SECOND(t)	secondi;
STR_TO_DATE(s, f)	converte una stringa in una data base al formato *f*;
SYSDATE()	data e ora corrente;
YEAR(d)	anno.

```
mysql> SELECT DATEDIFF('2009-04-18', '2009-01-27');
+-------------------------------------+
| DATEDIFF('2009-04-18', '2009-01-27') |
+-------------------------------------+
|                                  81 |
+-------------------------------------+

mysql> SELECT DATE_ADD('2009-12-31 22.00.00', INTERVAL 12 hour);
+---------------------------------------------------+
| DATE_ADD('2009-12-31 22.00.00', INTERVAL 12 hour) |
+---------------------------------------------------+
| 2010-01-01 10:00:00                               |
+---------------------------------------------------+

mysql> SELECT DAY('2010-12-31') AS Giorno,
    ->        MONTHNAME('2010-12-31') AS MESE,
    ->        YEAR('2010-12-31') AS ANNO,
    ->        DAYNAME('2010-12-31') AS '',
    ->        DAYOFYEAR('2010-12-31') AS Numero;
+--------+----------+------+--------+--------+
| Giorno | MESE     | ANNO |        | Numero |
+--------+----------+------+--------+--------+
|     31 | December | 2010 | Friday |    365 |
+--------+----------+------+--------+--------+
```

Funzioni per la manipolazione di stringhe

CONCAT(s1,s2,...)	concatena le stringhe;
INSERT(s1, p, n, s2)	sostituisce in *s1*, a partire da *p*, *n* caratteri di *s2*;
LCASE(s) o LOWER(s)	converte tutti i caratteri in minuscolo;
LEFT(s,n)	restituisce gli *n* caratteri più a sinistra di *s*;
LENGTH(s)	lunghezza della stringa;

Loescher Editore - Vietata la vendita e la diffusione

LOCATE(sottostringa, s)	individua la posizione della prima occorrenza di *sottostringa* in *s*;
REPLACE(s, x, y)	sostituisce nella stringa *s* tutte le occorrenze di *x* con *y*;
REVERSE(s)	inverte la stringa;
RIGHT(s,n)	restituisce gli *n* caratteri più a destra di *s*;
STRCMP(s1,s2)	confronta le stringhe e restituisce un numero <, = o > di 0;
SUBSTR(s, p, n)	restituisce la sottostringa di n caratteri che inizia alla posizione *p* di *s*;
TRIM(s)	toglie tutti gli spazi iniziali e finali;
UCASE() o UPPER()	converte tutti i caratteri in maiuscolo.

```
mysql> SELECT INSERT('ciao a tutti', 8, 5, 'voi');
+-------------------------------------+
| INSERT('ciao a tutti', 8, 5, 'voi') |
+-------------------------------------+
| ciao a voi                          |
+-------------------------------------+

mysql> SELECT LENGTH('abracadabra'), LEFT('abracadabra', 4);
+-----------------------+------------------------+
| LENGTH('abracadabra') | LEFT('abracadabra', 4) |
+-----------------------+------------------------+
|                    11 | abra                   |
+-----------------------+------------------------+

mysql> SELECT REPLACE('abracadabra','a','o'),
REVERSE('abracadabra');
+--------------------------------+------------------------+
| REPLACE('abracadabra','a','o') | REVERSE('abracadabra') |
+--------------------------------+------------------------+
| obrocodobro                    | arbadacarba            |
+--------------------------------+------------------------+
```

Altre funzioni

IF(e1,e2,e3)	valuta la prima espressione e se è vera restituisce la seconda altrimenti la terza;
GREATEST(v1,v2,...)	restituisce il più grande dei valori;
LEAST(v1,c2,...)	restituisce il più piccolo dei valori.

```
mysql> SELECT IF (5>3, 'vero', 'falso');
+---------------------------+
| IF (5>3, 'vero', 'falso') |
+---------------------------+
| vero                      |
+---------------------------+

mysql> SELECT GREATEST('cane', 'gatto', 'ape', 'cavallo');
+---------------------------------------------+
| GREATEST('cane', 'gatto', 'ape', 'cavallo') |
+---------------------------------------------+
| gatto                                       |
+---------------------------------------------+
```

Loescher Editore - Vietata la vendita e la diffusione

Comandi

La sintassi dei comandi di definizione dei dati, manipolazione e interrogazione in generale segue lo standard SQL; esistono però alcune clausole aggiuntive ed alcune limitazioni proprie di MySQL.

Per esempio MySQL non supporta la definizione di vincoli intrarelazionali tramite la clausola CHECK di CREATE TABLE.

Nota: Per specificare alcuni vincoli intrarelazionali si possono usare le viste o i trigger.

MySQL supporta il comando UNION (solo tra query e non tra tabelle), ma non i comandi INTERSECT ed EXCEPT.

7.3 DEFINIZIONE DEI DATI

Nota: Per eseguire queste operazioni bisogna avere i permessi adeguati, per esempio essere collegati come root.

SHOW CREATE TABLE

SHOW CREATE TABLE nomeTabella;

visualizza il comando necessario per la creazione di una tabella esistente.

Motori di memorizzazione

Il **motore di memorizzazione** (**storage engine**) è il sistema usato dal DBMS per salvare i dati su disco.
Esistono diversi motori di memorizzazione; i principali sono *MyISAM* e *InnoDB*.
MyISAM è lo storage engine predefinito per le nuove tabelle (se non è specificato diversamente) ed è molto efficiente, in particolare per la lettura e l'inserimento dei dati.
InnoDB non offre le stesse prestazioni di MyISAM, ma supporta i campi auto increment (contatori), le chiavi esterne e le transazioni e offre una migliore procedura di recupero in caso di crash del sistema; per sfruttarne le caratteristiche bisogna specificare esplicitamente che si desidera creare una tabella di tipo *InnoDB*.

```
CREATE TABLE Tabella (
    ...
) ENGINE=InnoDB;
```

Campi contatore

È possibile definire un campo **contatore** con la clausola **AUTO_INCREMENT**.
Può esserci un solo campo AUTO_INCREMENT per tabella, che deve essere numerico senza segno, senza valore di default e deve essere un campo indice (se è una chiave lo è automaticamente).
Il valore di un campo contatore parte da 1 (se non specificato diversamente con la clausola AUTO_INCREMENT=n posta dopo la definizione dei campi) e viene incrementato automaticamente ad ogni inserimento.
Si può ottenere l'ultimo valore generato con la funzione LAST_INSERT_ID().

Durante l'inserimento il valore del campo contatore viene generato automaticamente se non è specificato o se è specificato NULL o 0 (ma può essere anche specificato esplicitamente).

Loescher Editore - Vietata la vendita e la diffusione

Esempio 7.2

Creazione di una tabella con un campo contatore e inserimento di alcuni dati.

```
mysql> CREATE TABLE Citta(
    -> IdCitta INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    -> NomeC VARCHAR(20) NOT NULL
    -> );

mysql> DESCRIBE Citta;
+---------+-----------------+------+-----+---------+----------------+
| Field   | Type            | Null | Key | Default | Extra          |
+---------+-----------------+------+-----+---------+----------------+
| IdCitta | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| NomeC   | varchar(20)     | NO   |     | NULL    |                |
+---------+-----------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

mysql> INSERT INTO Citta (NomeC) VALUES ('Roma');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Citta
    -> VALUES (NULL, 'Milano'),(0,'Torino'),(6,'Padova');
Query OK, 3 rows affected (0.00 sec)

mysql> SELECT * FROM Citta;
+---------+--------+
| IdCitta | NomeC  |
+---------+--------+
|       1 | Roma   |
|       2 | Milano |
|       3 | Torino |
|       6 | Padova|
+---------+--------+
4 rows in set (0.00 sec)
```

Indici

Gli **indici** (anche quelli creati automaticamente per le chiavi primarie e le chiavi esterne) possono essere definiti su campi di qualsiasi tipo, tranne quelli di tipo TEXT o BLOB.

Per i campi su cui è stato creato un indice, il comando **DESCRIBE** mostra nella colonna KEY i valori:

PRI	chiave primaria;
MUL	indice che permette valori duplicati;
UNI	indice che non permette valori duplicati.

Chiavi esterne e integrità referenziale

In MySQL le **chiavi esterne** devono essere definite con la clausola:

FOREIGN KEY (*Attributo1*, ... , *AttributoN*)
REFERENCES *Tabella* (*Attributo1*, ... , *AttributoN*)

Loescher Editore - Vietata la vendita e la diffusione

posta dopo la dichiarazione degli attributi, anche se la chiave esterna è composta da un solo attributo.

Nota

In MySQL la definizione inline della chiave esterna viene ignorata.

Il campo chiave esterna viene indicizzato automaticamente e deve fare riferimento a un campo su cui è definito (automaticamente o esplicitamente) un indice.

Per stabilire vincoli di integrità referenziale le tabelle devono usare lo storage engine InnoDB. Lo storage engine può essere specificato con la clausola ENGINE alla fine della dichiarazione (nelle versioni meno recenti viene usata la clausola TYPE).

Per una tabella già esistente si può modificare il tipo o aggiungere una chiave esterna con il comando ALTER TABLE.

Esempio 7.3

Creazione di chiavi esterne.

```
CREATE TABLE Litorale (
  Nazione varchar(20),
  Mare varchar(20),
  PRIMARY KEY (Nazione,Mare),
  FOREIGN KEY (Nazione) REFERENCES Nazioni (NomeN),
  FOREIGN KEY (Mare) REFERENCES Mari (NomeM)
) ENGINE=InnoDB;

CREATE TABLE Citta(
    IdCitta INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    NomeC VARCHAR(20) NOT NULL,
    Nazione VARCHAR(20) NOT NULL,
    Popolazione INT,
) ENGINE=InnoDB;

ALTER TABLE Citta
    ADD FOREIGN KEY Nazione REFERENCES Nazioni (NomeN)
    ON DELETE SET NULL ON UPDATE CASCADE;
```

Tabelle temporanee

In MySQL si possono creare tabelle temporanee con il comando **CREATE TEMPORARY TABLE**.

Nota

La clausola SELECT INTO è usata invece per esportare dati su file o variabili.

Esempio 7.4

Creazione di una tabella temporanea con il nome e la popolazione delle città italiane (esempio 3.13).

```
CREATE TEMPORARY TABLE PopolazioniCitta AS
    SELECT NomeC, Popolazione
    FROM Citta;
```

Loescher Editore - Vietata la vendita e la diffusione

7.4 MANIPOLAZIONE DEI DATI

Output dei comandi di manipolazione

Quando si esegue un comando di manipolazione dati viene restituito il numero di righe elaborate. Se il comando non va a buon fine vengono segnalati gli errori.
Si ottiene un errore se si tenta di inserire un record che non soddisfa eventuali vincoli, come per esempio quelli di integrità referenziale.

Esempio 7.5

Esempi di inserimenti di dati.

```
mysql> INSERT INTO Fiumi (NomeF,Lunghezza)
    -> VALUES ('Tamigi',346),('Loira',1013);
Query OK, 2 rows affected (0.00 sec)

mysql> INSERT INTO Litorale
    -> VALUES ('Spagna', 'Mar Mediterraneo');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key
constraint fails ('prove/litorale', CONSTRAINT 'litorale_ibfk_1'
FOREIGN KEY ('Nazione') REFERENCES 'nazioni' ('NomeN'))
```

Caricamento di dati da un file di testo

È possibile inserire in una tabella i dati memorizzati in un file di testo con il comando:

LOAD DATA INFILE *NomeFile*
INTO TABLE *Tabella*
[FIELDS TERMINATED BY 'carattere']
[FIELDS ENCLOSED BY 'carattere']
[LINES TERMINATED BY '\r\n'];

I record devono essere scritti nel file uno per riga; se il file di testo è scritto con un editor che termina le righe con i caratteri return e new line (\r e \n) è necessaria la clausola LINES TERMINATED BY.
Il separatore di default dei campi è il carattere di tabulazione; si può specificare un carattere diverso con la clausola FIELDS TERMINATED BY; le stringhe possono essere delimitate dal carattere specificato nella clausola FIELDS ENCLOSED BY e il valore NULL può essere rappresentato da \N.

Se il file non si trova nella directory corrente e bisogna specificare il path, il carattere backslash \ deve essere sostituito dalla sequenza di escape \\.

Nota Il file deve trovarsi sul server; per caricare un file che risiede sul client si deve usare la parola chiave LOCAL prima di INFILE.

7.5 INTERROGAZIONI

Output delle query

Il risultato di una query di selezione viene riportato in forma tabellare con i nomi dei campi come intestazioni delle colonne.

Loescher Editore - Vietata la vendita e la diffusione

Se la tabella ha molti campi o i record sono piuttosto lunghi è preferibile terminare Il comando SELECT con \G per ottenere l'output su righe successive.

Con entrambi i formati si può limitare il numero di record visualizzati con la clausola LIMIT.

Esempio 7.6

Visualizzazione dei soli primi due record prodotti da una query con l'output prima in forma tabellare e poi su righe successive.

```
mysql> SELECT NomeN AS Nazione, Superficie, Nazioni.Popolazione,
NomeCon AS Continente, NomeC AS Capitale, Citta.Popolazione FROM
Nazioni
    -> JOIN Continenti ON Continente=Sigla
    -> JOIN Citta ON Capitale=IdCitta
    -> LIMIT 2;
+---------+-----------+-------------+-----------+----------+-------------+
| Nazione | Superficie | Popolazione | Continente | Capitale | Popolazione |
+---------+-----------+-------------+-----------+----------+-------------+
| Egitto  |     998002 |    69537000 | Africa    | Il Cairo |     6500000 |
| Austria |      83871 |     8118000 | Europa    | Vienna   |     1550000 |
+---------+-----------+-------------+-----------+----------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT NomeN AS Nazione, Superficie, Nazioni.Popolazione,
NomeCon AS Continente, NomeC AS Capitale, Citta.Popolazione
    -> FROM Nazioni
    -> JOIN Continenti ON Continente=Sigla
    -> JOIN Citta ON Capitale=IdCitta
    -> LIMIT 2 \G
*************************** 1. row ***************************
    Nazione: Egitto
 Superficie: 998002
Popolazione: 69537000
 Continente: Africa
   Capitale: Il Cairo
Popolazione: 6500000
*************************** 2. row ***************************
    Nazione: Austria
 Superficie: 83871
Popolazione: 8118000
 Continente: Europa
   Capitale: Vienna
Popolazione: 1550000
2 rows in set (0.00 sec)
```

Esportazione di dati

È possibile memorizzare l'**output** di una query in un **file** con il comando:

SelectSQL
INTO OUTFILE 'nomeFile';

Per default i record vengono scritti nel file uno per riga e i campi sono separati da tabulazioni, ma è possibile cambiare questo comportamento.

Nota

Il file viene generato sul server con i permessi dell'utente associato al server MySQL e non quelli dell'utente che sta eseguendo il comando. Se il file esiste già, l'operazione viene interrotta.

Variabili

Nei comandi SELECT è possibile utilizzare delle **variabili**.

È possibile creare delle variabili e assegnare un valore, o modificarlo, con il comando SET.

SET @ *nomeVariabile* = *espressione*;

Il tipo della variabile dipende dal valore assegnato; se non viene assegnato nessun valore al variabile ha il valore NULL.

espressione può essere anche un comando SELECT racchiuso tra parentesi tonde.

È possibile anche assegnare un valore a una variabile all'interno di un comando SELECT usando **:=** come operatore di assegnazione.

Si può usare anche la sintassi:

SELECT *espressione* **INTO** *@variabile1* [, *@variabile2*...]
...

ma la query deve restituire una sola riga e il numero di attributi specificati non deve essere superiore al numero delle variabili.

Esempio 7.7

Alcuni esempi di uso di variabili.

```
mysql> SET @X=0;
mysql> SELECT @X;
+------+
| @X   |
+------+
| 0    |
+------+

mysql> SET @NumContinenti = (SELECT COUNT(*) FROM Continenti);

mysql> SELECT @NumContinenti;
+----------------+
| @NumContinenti |
+----------------+
| 7              |
+----------------+

mysql> SELECT @NumContinenti := COUNT(*) FROM Continenti;
+----------------------------+
| @NumContinenti := COUNT(*) |
+----------------------------+
|                          7 |
+----------------------------+
```

Loescher Editore - Vietata la vendita e la diffusione

```
mysql> SELECT NomeC, Popolazione INTO @NomeCitta, @Popolazione
    -> FROM Citta
    -> WHERE idCitta = 11;

mysql> select @NomeCitta,@Popolazione;
+------------+--------------+
| @NomeCitta | @Popolazione |
+------------+--------------+
| Vienna     | 1550000      |
+------------+--------------+
```

Query parametriche

Si possono usare le variabili per definire **query parametriche**.

Esempio 7.8

Nome e popolazione delle città di una nazione specificata come parametro.

```
mysql> SET @Nazione='Italia';
mysql> SELECT NomeC, Popolazione FROM Citta
    -> WHERE Nazione=@Nazione;
```

Nome e popolazione delle città che appartengono a una nazione specificata come parametro e che hanno popolazione superiore a un valore specificato come parametro.

```
mysql> SET @Nazione='Italia';
mysql> SET @Pop=1000000;
mysql> SELECT NomeC, Popolazione FROM Citta
    -> WHERE Nazione=@Nazione AND Popolazione>@Pop;
```

Le query possono essere memorizzate in un **file** di testo e richiamate quando desiderato con il comando **SOURCE**.

```
mysql> SET @Nazione='Italia';
mysql> SOURCE Query.sql;
```

Quando si memorizzano le query nei file è bene inserire dei commenti esplicativi.
I commenti compresi tra /* e */ possono estendersi su più righe; i commenti che iniziano con -- terminano alla fine della riga.

7.6 VISTE

In MySQL nella creazione delle **viste** è disponibile la clausola **WITH CHECK OPTION** che permette di definire vincoli intrarelazionali.
Inserendo i dati mediante la vista anziché direttamente è possibile effettuare dei controlli sui dati inseriti nel database.

Esempio 7.9

Creare una vista per controllare che la lunghezza dei fiumi non possa essere superiore a 6671 km, lunghezza del Nilo (il fiume più lungo).

Loescher Editore - Vietata la vendita e la diffusione

```
mysql> CREATE VIEW ControlloFiumi AS
    -> SELECT *
    -> FROM Fiumi
    -> WHERE Lunghezza < 6671
    -> WITH CHECK OPTION;

mysql> INSERT INTO ControlloFiumi VALUES ('Loira',12000);
ERROR 1369 (HY000): CHECK OPTION failed 'geografia.ControlloFiumi'
```

COSTRUTTI PER IL CONTROLLO DI FLUSSO

MySQL supporta vari costrutti per il **controllo di flusso**:

IF *condizione* **THEN**
...
ELSE
...
END IF;

WHILE *condizione* **DO**
...
END WHILE;

CASE *espressione*
WHEN *valore1* **THEN** ...
[WHEN *valore2* THEN ...] ...
[**ELSE** ...]
END CASE

REPEAT
...
UNTIL *condizione*
END REPEAT;

Nota I cicli FOR non sono supportati.

I costrutti per il controllo di flusso possono essere usati nella definizione di trigger, funzioni e procedure.

7.7 TRIGGER

Nota I trigger sono supportati dalla versione 5.0 di MySQL.

Il comando per la definizione di trigger è:

CREATE TRIGGER *NomeTrigger Attivazione Evento*
ON *Tabella*
FOR EACH ROW
Corpo;

NomeTrigger è il nome del trigger, che deve essere unico per la tabella specificata; il trigger è associato alla tabella quindi ha il nome *Tabella.NomeTrigger*;

Attivazione può assumere i valori **BEFORE** o **AFTER** e indica se il trigger deve essere attivato prima o dopo l'evento specificato;

Evento specifica l'evento che innesca l'attivazione del trigger; i valori possibili sono **INSERT**, **DELETE** e **UPDATE** e non si riferiscono solo ai comandi corrispondenti ma anche ad altri comandi che hanno lo stesso effetto (per esempio l'evento INSERT si verifica anche nel caso di inserimento di dati con LOAD DATA);

Loescher Editore - Vietata la vendita e la diffusione

ATTENZIONE

Non possono esserci due trigger con lo stesso momento di attivazione e lo stesso evento.

Tabella indica il nome della tabella a cui è associato il trigger; deve essere una tabella reale (non può essere una tabella temporanea o una vista);

Corpo è l'azione che deve essere svolta quando il trigger è innescato; se ci sono più istruzioni devono essere racchiuse tra BEGIN e END e in tal caso è necessario cambiare il carattere di terminazione con il comando DELIMITER in modo da poter usare il ; per terminare le istruzioni all'interno della definizione del trigger.

Nei trigger si possono usare comandi SQL ma il comando SELECT deve essere nel formato:

SELECT *espressione* INTO *@variabile1* [, *@variabile2*...] FROM ...

Inoltre non si può modificare una tabella che è già stata usata in lettura o in scrittura dal comando che ha invocato il trigger.

Un trigger può accedere ai dati della tabella a cui è associato solo tramite gli alias **OLD** e **NEW**: *OLD.Attributo* indica il valore dell'attributo prima dell'evento che attiva il trigger, *NEW.Attributo* il valore dopo l'evento. *OLD.Attributo* non può mai essere modificato; *NEW.Attributo* può essere modificato solo in trigger con attivazione BEFORE.

Viene restituito un errore se si usa NEW per effettuare modfiche in un trigger di tipo AFTER:
`ERROR 1362 (HY000): Updating of NEW row is not allowed in after trigger.`

Si può eliminare un trigger col comando:

DROP TRIGGER NomeTrigger;

ESEMPIO 7.10

Controllo che nell'inserimento di una nuova città la popolazione della città non superi la popolazione della nazione; se ciò si verifica viene impostata a NULL la popolazione della città.

Nota Si dovrebbe definire un trigger analogo anche per l'evento UPDATE.

```
mysql> DELIMITER //

mysql> CREATE TRIGGER PopolazioneCitta BEFORE INSERT ON Citta
    -> FOR EACH ROW
    ->     BEGIN
    ->         SELECT Popolazione INTO @popNazione
    ->         FROM Nazioni
    ->         WHERE NomeN = NEW.Nazione;
    ->
    ->         IF NEW.Popolazione > @popNazione THEN
    ->             SET NEW.Popolazione = NULL;
    ->         END IF;
    -> END //
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;

mysql> INSERT INTO Citta (NomeC, Nazione, Popolazione) VALUES
('Venezia', 'Italia', 70000000);
Query OK, 1 row affected (0.01 sec)
```

Loescher Editore - Vietata la vendita e la diffusione

```
mysql> SELECT * FROM Citta WHERE NomeC='Venezia';
+---------+---------+---------+-------------+
| idCitta | NomeC   | Nazione | Popolazione |
+---------+---------+---------+-------------+
|      24 | Venezia | Italia  |        NULL |
+---------+---------+---------+-------------+
```

Esempio 7.11

Controllo che nell'inserimento di *Presenza (Nazione, Lago)* la superficie del lago non renda complessivamente superiore la somma della superficie dei laghi della nazione alla superficie della nazione; se ciò si verifica viene impostata a NULL la superficie del lago.

```
CREATE TRIGGER SuperficieLago AFTER INSERT ON Presenza
FOR EACH ROW
   BEGIN
      SELECT SUM(Superficie) INTO @superficieLaghi
         FROM Laghi
         JOIN Presenza ON Lago = NomeL
         WHERE Nazione = NEW.Nazione;

      SELECT Superficie INTO @superficieNazione
          FROM Nazioni
          WHERE NomeN = NEW.Nazione;

      IF @superficieLaghi > @superficieNazione THEN
         UPDATE Laghi SET Superficie = NULL WHERE NomeL = NEW.Lago;
      END IF;
   END //
```

I trigger possono essere utilizzati per **impedire il completamento** di un'operazione mediante opportuni accorgimenti.
Per esempio per impedire l'inserimento di un record in una tabella si può predisporre nella tabella un record con un valore predefinito in un campo UNIQUE; per impedire l'inserimento di un nuovo record basta modificare il valore del campo corrispondente sostituendolo con quello già presente, in questo modo l'inserimento del record causa un errore, dato che non possono esistere due record con lo stesso valore, bloccando l'operazione.

Esempio 7.12

Controllo che nell'inserimento di una nuova città la popolazione della città non superi la popolazione della nazione; se ciò si verifica si impedisce l'inserimento del record.

```
CREATE TRIGGER PopolazioneCitta BEFORE INSERT ON Citta
FOR EACH ROW
    BEGIN
        SELECT Popolazione INTO @popNazione
        FROM Nazioni
        WHERE NomeN = NEW.Nazione;

        IF NEW.Popolazione > @popNazione THEN
            SET NEW.NomeC = 'NonValido';
        END IF;
END //
```

Loescher Editore - Vietata la vendita e la diffusione

```
mysql> INSERT INTO Citta (NomeC, Nazione, Popolazione) VALUES
('Venezia', 'Italia', 70000000);
ERROR 1062 (23000): Duplicate entry 'NonValido-Italia' for key 2
```

7.8 STORED ROUTINE

Le **stored routine** sono funzioni o procedure memorizzate nel database (e quindi utilizzabili sono nel database in cui sono state definite). Le funzioni restituiscono un valore, mentre le procedure per restituire valori usano i parametri.

Nota: Le stored routine sono supportate dalla versione 5.0 di MySQL.

FUNZIONI

Il comando per la creazione di funzioni è:

CREATE FUNCTION *NomeFunzione* ([*Parametri*])
RETURNS *Tipo*
Corpo;

NomeFunzione è il nome della funzione;

Parametri sono i parametri di input; vanno indicati col nome seguito dal tipo MySQL e vanno separati da virgole;

Tipo è il tipo MySQL del valore restituito dalla funzione;

Corpo è il corpo della funzione; se ci sono più istruzioni devono essere racchiuse tra BEGIN e END e in tal caso è necessario cambiare il carattere di terminazione con il comando DELIMITER in modo ad poter usare il ; per terminare le istruzioni all'interno della definizione della funzione.

Le funzioni devono contenere una istruzione **RETURN** per restituire un valore e possono usare variabili locali dichiarate col comando:

DECLARE *nomeVariabile tipoVariabile*;

Nelle funzioni si possono usare comandi SQL ma il comando SELECT deve essere nel formato:

SELECT *espressione* INTO *variabile1* [, *variabile2*...] FROM ...

Inoltre non si può modificare una tabella che è già stata usata in lettura o in scrittura dal comando che ha invocato la funzione.

Nota: Funzioni e trigger hanno le stesse limitazioni.

Per richiamare una funzione basta usare il nome della funzione seguito dai parametri tra parentesi, separati da virgole; se la funzione non richiede parametri bisogna comunque usare le parentesi.

Si può eliminare una funzione col comando

DROP FUNCTION nomeFunzione;

ESEMPIO 7.13
Funzione che trasforma una data dal formato usato da MySQL *aaaa-mm-gg* nel formato *gg/mm/aa*.

Loescher Editore - Vietata la vendita e la diffusione

```
mysql> DELIMITER //

mysql> CREATE FUNCTION DataFormattata(DataMySQL DATE)
    -> RETURNS VARCHAR(8)
    -> RETURN DATE_FORMAT(DataMySQL,'%d/%m/%y');
    -> //

mysql> DELIMITER ;

mysql> SELECT CURDATE() AS 'Data MySQL', DataFormattata(CURDATE())
AS 'Data formattata';
+------------+-----------------+
| Data MySQL | Data formattata |
+------------+-----------------+
| 2008-11-23 | 23/11/08        |
+------------+-----------------+
```

Esempio 7.14

Funzione che conta le città di una data nazione.

```
CREATE FUNCTION ContaCitta( NazioneObiettivo VARCHAR(100) )
    RETURNS INT
    BEGIN
       DECLARE Conta INT;
       SELECT COUNT(*) INTO Conta
       FROM Citta
          WHERE Nazione=NazioneObiettivo;
          RETURN Conta;
    END //

mysql> SELECT ContaCitta('Italia') AS NCittaItaliane;
+----------------+
| NCittaItaliane |
+----------------+
|              5 |
+----------------+
```

Procedure

Il comando per la definizione di procedure è:

CREATE PROCEDURE *NomeProcedura* ([*Parametri*])
Corpo

NomeProcedura è il nome della procedura;

Parametri sono i parametri di input, output e input/output; vanno indicati col nome preceduto dalla direzione (**IN** per input, **OUT** per output e **INOUT** per input/output) e seguito dal tipo MySQL e vanno separati da virgole;

Corpo è il corpo della procedura; se ci sono più istruzioni devono essere racchiuse tra BEGIN e END e in tal caso è necessario cambiare il carattere di terminazione con il comando DELIMITER in modo ad poter usare il ; per terminare le istruzioni all'interno della definizione della procedura.

Le procedure possono contenere qualunque tipo di comando SQL (anche comandi SELECT che restituiscono tabelle e non solo valori scalari).

Loescher Editore - Vietata la vendita e la diffusione

Il comando per il richiamo di una procedura è:

CALL *NomeProcedura* ([*Parametri*]);

i parametri vanno separati da virgole; se la procedura non richiede parametri bisogna comunque usare le parentesi.
Per i parametri di input come parametri attuali si possono passare costanti o variabili, mentre per parametri di output o di input/output solo variabili.

Si può eliminare una procedura col comando:

DROP PROCEDURE nomeProcedura;

Esempio 7.15

Inserimento di una nuova città con controllo che nell'inserimento di una nuova città la popolazione della città non superi la popolazione della nazione; se ciò si verifica si impedisce l'inserimento del record.

```
mysql> DELIMITER //

mysql> CREATE PROCEDURE InserisciCitta (IN NomeCitta VARCHAR(100),
IN NomeNazione VARCHAR(50), IN Pop INT)
    -> BEGIN
    ->     SELECT Popolazione INTO @popNazione
    ->     FROM Nazioni
    ->     WHERE NomeN = NomeNazione;
    ->
    ->     IF Pop > @popNazione THEN
    ->         SET @esito = 'Errore: popolazione della Citta`
troppo elevata';
    ->     ELSE
    ->         INSERT INTO Citta (NomeC, Nazione, Popolazione)
VALUES (NomeC, NomeNazione, Pop);
    ->         SET @esito = 'Inserimento avvenuto con successo';
    ->     END IF;
    ->
    ->     SELECT @esito AS 'Esito inserimento';
    ->
    -> END //
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;

mysql> CALL InserisciCitta('Genova','Italia',70000000);
+-------------------------------------------------+
| Esito inserimento                               |
+-------------------------------------------------+
| Errore: popolazione della citta` troppo elevata |
+-------------------------------------------------+

mysql> CALL InserisciCitta('Genova','Italia',611000);
+-----------------------------------+
| Esito inserimento                 |
+-----------------------------------+
| Inserimento avvenuto con successo |
+-----------------------------------+
```

Loescher Editore - Vietata la vendita e la diffusione

ESEMPIO 7.16

Inserimento di un nuovo lago e calcolo della superficie media dei laghi.

```
CREATE PROCEDURE InserisciLago (IN NomeLago VARCHAR(100), IN Sup
INT, OUT SommaSuperficiLaghi INT, OUT MediaSuperficiLaghi INT)
   BEGIN
      INSERT INTO Laghi (NomeL, Superficie) VALUES (NomeLago, Sup);
      SELECT SUM(Superficie) INTO SommmaSuperficiLaghi
      FROM Laghi;
      SELECT AVG(Superficie) INTO MediaSuperficiLaghi
      FROM Laghi;
   END //
```

```
mysql> CALL InserisciLago('Loch Ness',56,@SommaSuperf,@MediaSuperf);
```

```
mysql> SELECT @SommaSuperf, @MediaSuperf;
```

7.9 CONTROLLO DELL'ACCESSO (GRANT)

In MySQL la sintassi del comando GRANT è:

GRANT *Privilegi*
ON *Oggetto*
TO *Utente* [IDENTIFIED BY [PASSWORD] 'password']
[, *Utente* [IDENTIFIED BY [PASSWORD] 'password']...]
[WITH *Opzione* [*Opzione*...]]

I **privilegi** principali sono INSERT, DELETE, UPDATE, SELECT, CREATE, DROP e ALL.

oggetto può essere il nome di una tabella, il carattere speciale * (dopo aver scelto uno specifico database) o la stringa *nomeDatabase.** per indicare tutti gli elementi (tabelle e routine) di un database, la stringa *.* per indicare tutti gli elementi di tutti i database, oppure la stringa *nomeDatabase.nomeRoutine* per indicare una singola routine.

utente può essere indicato con la sintassi *nomeUtente@dominio* per limitare l'accesso da client remoti; il carattere speciale % rappresenta qualunque dominio.

La clausola IDENTIFIED BY permette di specificare una password per l'utente.

La clausola WITH permette di specificare opzioni aggiuntive per l'utente, come per esempio un numero massimo di query che l'utente può eseguire ogni ora.

7.10 TRANSAZIONI

In MySQL una transazione utilizza la sintassi:

SET AUTOCOMMIT=0;
START TRANSACTION;
INSERT ...
UPDATE ...
COMMIT;
SET AUTOCOMMIT=1;

Loescher Editore - Vietata la vendita e la diffusione

La transazione inizia impostando AUTOCOMMIT a 0 in modo da richiedere che i comandi non vengano eseguiti subito, ma soltanto dopo il comando COMMIT.
Al termine AUTOCOMMIT deve essere riportato al valore di default.

Esempio 7.17

Se una nazione viene divisa in due nazioni la transazione comprende la creazione della nuova nazione e la variazione dei dati della prima nazione.

```
SET AUTOCOMMIT=0;
START TRANSACTION;
INSERT INTO Nazioni (NomeN,Superficie,Popolazione) VALUES
('Slovacchia',50000,5000000);
UPDATE Nazioni SET NomeN='Rep. Ceca', Superficie=80000,
Popolazione=10000000 WHERE NomeN='Cecoslovacchia';
COMMIT;
SET AUTOCOMMIT=1;
```

Se il database contiene un record per la Cecoslovacchia. **Nota**

MYSQL GUI TOOLS

Per interagire e amministrare il database si possono usare programmi a interfaccia grafica. ***MySQL GUI Tools***, un pacchetto gratuito (rilasciato sotto licenza GNU/Gpl) distribuito dallo stesso team che si occupa dello sviluppo di MySQL, contiene ***MySQL Query Browser*** per interagire col database e ***MySQL Administrator*** per effettuare operazioni di amministrazione.

7.11 MYSQL QUERY BROWSER

La connessione

All'avvio ***MySQL Query Browser*** presenta una finestra per l'inserimento dei parametri di connessione.

In particolare sono richiesti l'indirizzo (*Server Host*) e la porta del server MySQL, nome utente e password; si può specificare un database predefinito (*Default Schema*).

È possibile salvare delle connessioni, in modo da non dover inserire ogni volta tutti i parametri. **Nota**

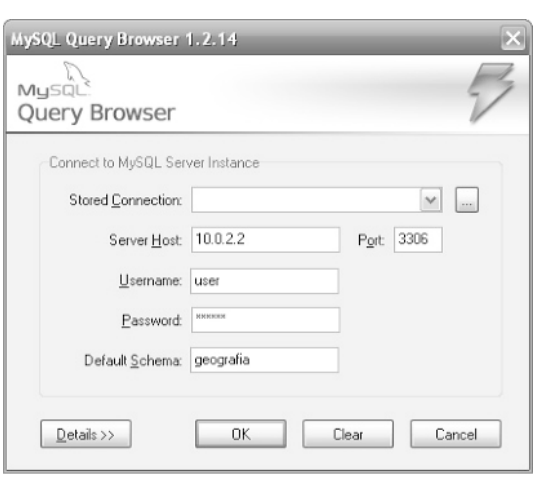


Figura 7.3 *Login al server MySQL da MySQL Query Browser*

La visibilità dei database e le operazioni disponibili una volta effettuato il login dipendono dai privilegi concessi all'utente con cui si accede. **Nota**

Loescher Editore - Vietata la vendita e la diffusione

Dopo aver effettuato il login è visualizzata la finestra principale del programma.

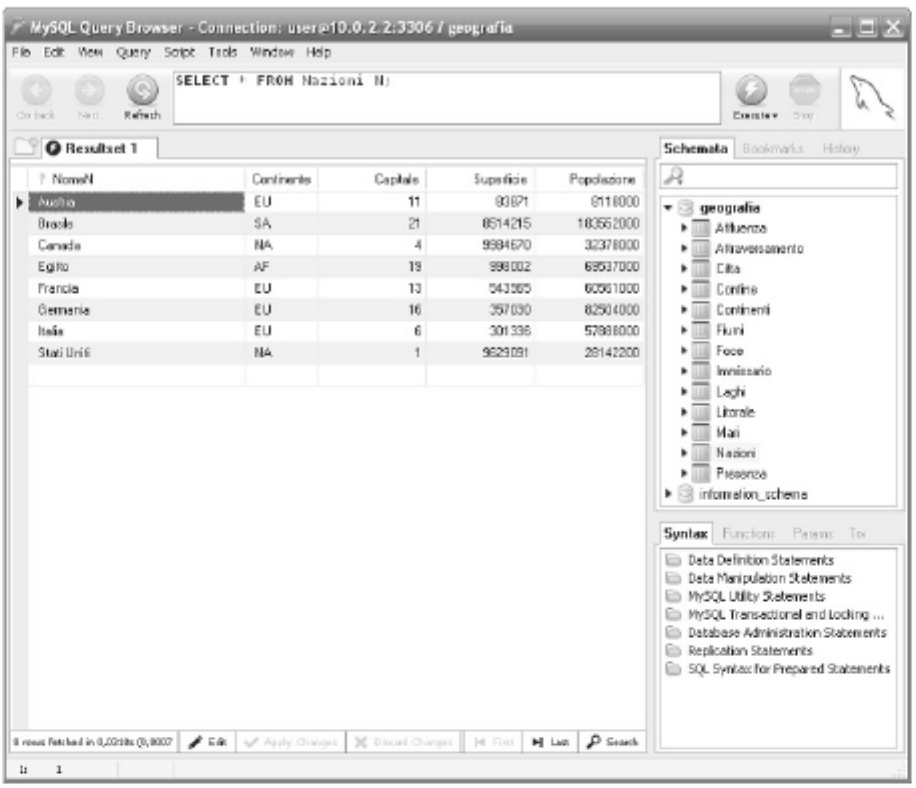


Figura 7.4 *Finestra principale di MySQL Query Browser*

L'editor di query

In alto si trova la barra dell'**editor di query**. Nell'area di testo (***Query area***) è possibile scrivere una query o comporla in modalità grafica. I pulsanti ***Go Back*** e ***Next*** servono per navigare tra le query eseguite recentemente, ***Refresh*** aggiorna il risultato della query, ***Execute*** esegue la query (la stessa azione si ottiene tramite la combinazione *Ctrl+Invio*) e ***Stop*** interrompe l'esecuzione.

Quando si trascina una tabella nell'area della query, sotto l'area viene presentata una barra di pulsanti di composizione; rilasciando la tabella su uno dei pulsanti si ottiene il seguente effetto:

Select	sostituisce la query corrente con il comando SELECT che visualizza i dati della tabella selezionata;
Add Table(s)	aggiunge la tabella nella clausola FROM della query;
Join Table(s)	aggiunge la tabella nella clausola FROM della query, e costruisce la clausola WHERE necessaria per il join (sfrutta le integrità referenziali, se ci sono, per individuare i campi comuni);
Left Outer Join	costruisce il join esterno sinistro tra la tabella che si trova già nell'editor e quella trascinata;
Insert	sostituisce la query corrente con il comando di inserimento di dati nella tabella trascinata;
Update	sostituisce la query corrente con il comando di modifica della tabella trascinata;
Delete	sostituisce la query corrente con il comando di cancellazione della tabella trascinata.

Figura 7.5 *Barra dei pulsanti di composizione*

Loescher Editore - Vietata la vendita e la diffusione

Il tasto ***F11*** o il comando di menu ***View***, ***Maximize Query Edit*** apre un editor di query più completo.

È possibile costruire automaticamente una query in modalità "point-and-clic" scegliendo la clausola (come SELECT, FROM, WHERE...) e la tabella o i campi desiderati.

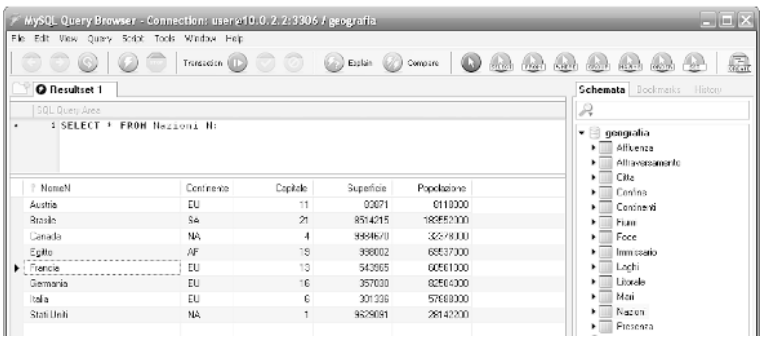


Figura 7.6 *Visualizzazione completa dell'editor di query*

L'area dei risultati

L'**area dei risultati** (***Result Area***) è divisa in schede, in modo da poter visualizzare contemporaneamente i risultati di più query.
Si può aprire una nuova scheda con il pulsante a sinistra delle schede, oppure tramite la combinazione *Ctrl+T*; per navigare tra le schede basta fare clic sul titolo della scheda opportuna, oppure usare le combinazioni *Ctrl+Tab* (avanti) e *Shift+Ctrl+Tab* (indietro).
Si può suddividere orizzontalmente o verticalmente la scheda dei risultati, per esempio per confrontare i risultati di diverse query, facendo clic destro in un punto qualsiasi dell'area e scegliendo *Split Tab Vertically* o *Split Tab Orizontally*.

La barra di stato

Se la query coinvolge una sola tabella è possibile modificare i dati attivando la funzione ***Edit*** col pulsante che si trova nella barra di stato.

Nota Se si eseguono join tra più tabelle o si lavora su viste, la funzione *Edit* non è disponibile.

Dopo aver attivato la funzione basta un doppio clic su una riga dei risultati per abilitare la modifica del contenuto; un doppio clic sulla riga vuota in fondo ai risultati permette di aggiungere un nuovo record; per cancellare un record basta fare clic destro sulla riga e scegliere *Delete Row(s)*.
Tutte le modifiche effettuate vengono apportate effettivamente al database solo quando si seleziona il pulsante ***Apply Changes*** sulla barra di stato.

I box a destra

Si possono salvare e riutilizzare query tramite le funzionalità ***History*** (che contiene tutte le query eseguite dall'utente di recente) e ***Bookmarks*** accessibili dal box a destra nella finestra principale del programma.

La scheda ***Schemata*** del box a destra permette di selezionare database e tabelle.
Si può creare una nuova tabella facendo clic destro nell'area *Schemata* e scegliendo ***Create New Table***.
Si può modificare una tabella esistente facendo clic destro sulla tabella e scegliendo ***Edit Table***.
In entrambi i casi si accede alla finestra ***Table editor*** che presenta in alto le caselle per il nome della tabella e il database a cui appartiene; le schede in basso aiutano a definire la tabella: *Column Details* per i campi, *Indices* per gli indici e *Foreign Keys* per le chiavi esterne e i vincoli di integrità referenziali che la tabella deve soddisfare (solo se il motore di memorizzazione è InnoDB; il motore di memorizzazione può essere specificato nella scheda *Table Options*).
Per creare o modificare effettivamente la tabella bisogna selezionare il pulsante ***Apply Changes***.

Loescher Editore - Vietata la vendita e la diffusione

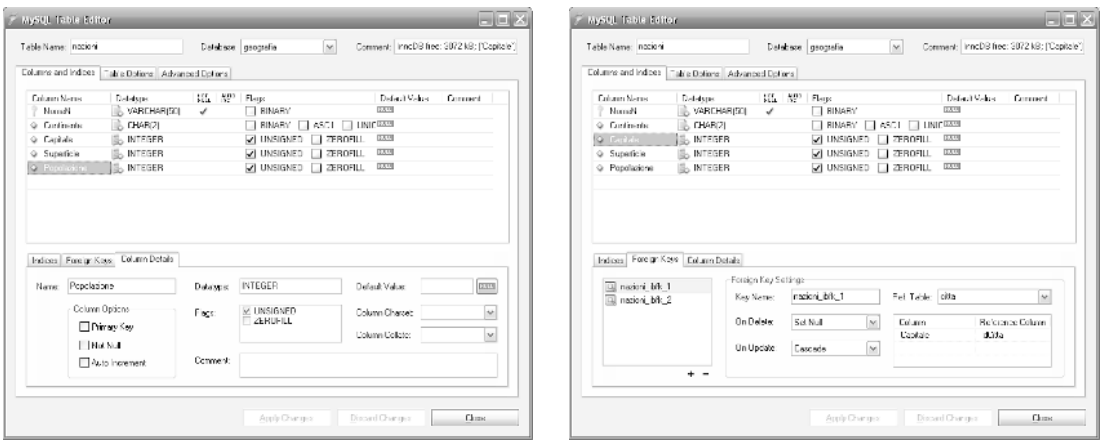


Figura 7.7 *Table editor, scheda Column Details e scheda Foreign Keys*

Dal menu di scelta rapida che si ottiene facendo clic destro nell'area *Schemata* sono disponibili anche i comandi ***Create New View*** per creare viste e ***Create New Procedure/Function*** per creare procedure e funzioni
Viste e procedure vengono elencate nell'area *Schemata*, subito dopo le tabelle.

Nota

Per creare una vista che si basa su una query si può usare il pulsante *Create* che compare nell'editor di query in visualizzazione completa.

Il box in basso a destra permette di consultare la **documentazione** di MySQL.

Esportazione

Il comando di menu ***File***, ***Export Resultset*** permette di esportare i dati nei formati HTML, XML, Microsoft Excel, CVS (Comma-separated values) e property list (PLIST, usato da vari linguaggi di programmazione, per esempio Java).

7.12 MYSQL ADMINISTRATOR

MySQL Administrator all'avvio presenta la finestra di connessione in cui inserire host, porta, nome utente e password.

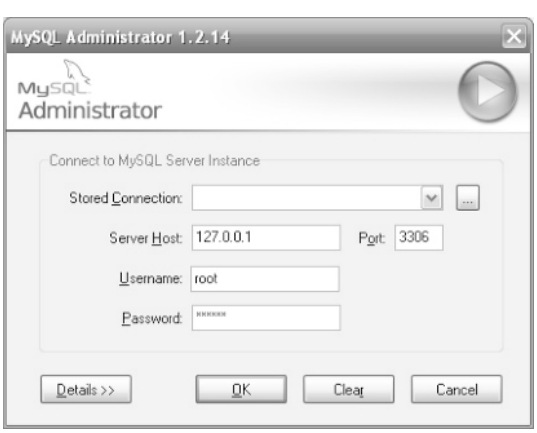


Figura 7.8
Login al server MySQL da MySQL Administrator

Dopo aver effettuato il login si accede alla finestra principale del programma; nel riquadro a sinistra si seleziona la funzione desiderata; le informazioni relative sono presentate nel riquadro a destra.

Informazioni, avvio e parametri

Server Information presenta informazioni sul server MySQL e sul client utilizzati.

Loescher Editore - Vietata la vendita e la diffusione

Nella sezione ***Service Control*** si può avviare o spegnere il server MySQL. Sono inoltre visualizzati i log di avvio e spegnimento, in modo che sia possibile identificare eventuali problemi (questa sezione è disponibile solo se *MySQL Administrator* viene avviato sullo stesso computer del server MySQL (host: localhost o 127.0.0.1).

La sezione ***Startup Variables*** permette di configurare diverse impostazioni del server MySQL; è possibile tra l'altro attivare o disattivare le connessioni remote via TCP/IP, cambiare la porta di ascolto del server, scegliere il percorso dei file di log e scegliere i motori di memorizzazione.

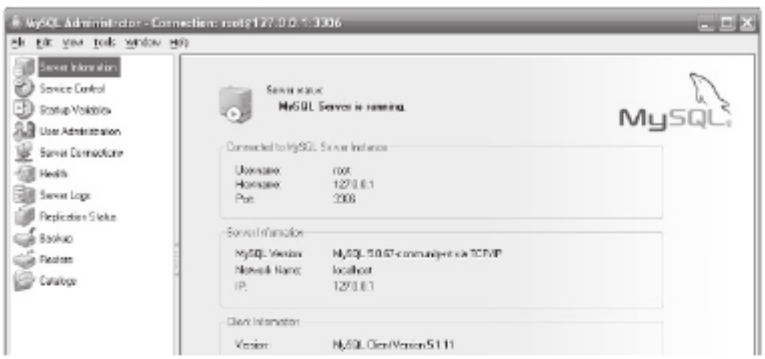


Figura 7.9 *Server Information*

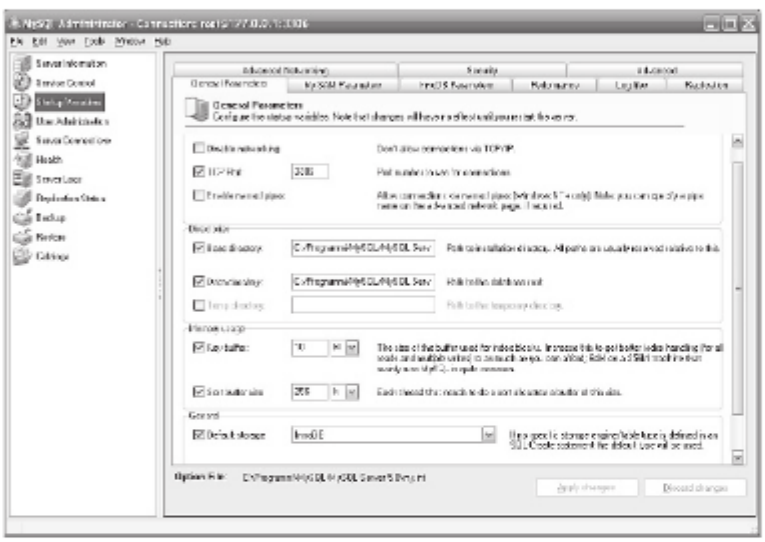


Figura 7.10 *Startup Variables*

Utenti e privilegi

User Administration permette di creare o rimuovere utenti e di assegnare o revocare privilegi.
Per creare un utente è sufficiente fare clic su *Add new user* e inserire alcune informazioni sull'utente: nome di login e password e informazioni aggiuntive come nome completo, indirizzo e-mail, descrizione e foto.

Per assegnare i privilegi dell'utente sui singoli database basta accedere alla scheda *Schema Privileges*, scegliere il database e spostare i permessi che si desidera assegnare dall'elenco *Available Privileges* a *Assigned Privileges*, tramite le frecce poste tra i due elenchi.

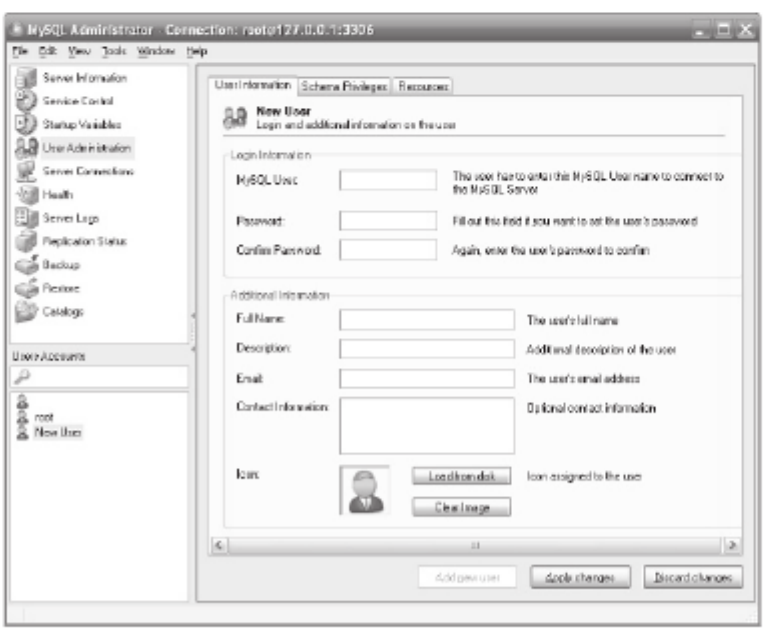


Figura 7.11 *User Administration*

Loescher Editore - Vietata la vendita e la diffusione

Funzioni di diagnostica

Le sezioni ***Server Connections***, ***Health*** e ***Server Logs*** sono utili per la diagnostica del sistema.

Server Connections visualizza informazioni sugli utenti connessi al server: nome utente, host dal quale si è connesso, stato attuale (è sleep se l'utente è connesso ma non sta eseguendo alcun comando) e perfino l'operazione che l'utente sta eseguendo.

La sezione ***Health*** è composta da diverse schede: *Connection Health* e *Memory Health* contengono dei grafici che mostrano il numero di connessioni e l'utilizzo della memoria da parte del server MySQL, mentre le schede *Status Variables* e *Server Variables* mostrano le impostazioni di configurazioni (che possono essere modificate dalla sezione *Startup Variables*).

Server Logs mostra il contenuto dei file di log più importanti (*Error Log* e *General Log*).

Backup e Restore

La sezione ***Backup*** permette di creare progetti di backup complessi.
Bisogna fare clic su *New Project* e scegliere un nome e i database di cui fare il backup (si possono scegliere anche singole tabelle, spuntando opportunamente le voci).
La scheda *Advanced Options* permette di scegliere il tipo di esecuzione del backup.

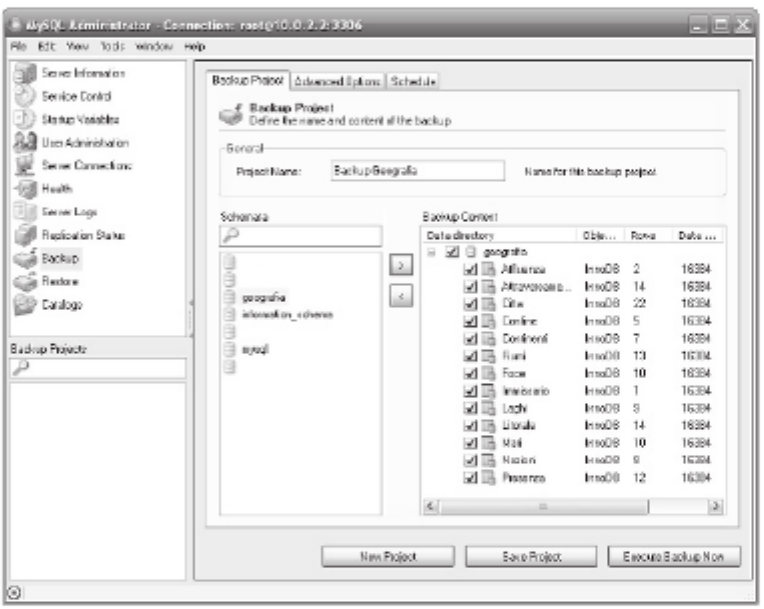


Figura 7.12
Backup

Nella scheda *Schedule* è possibile eseguire immediatamente il backup tramite il pulsante *Execute Backup Now* in basso a destra o pianificare il backup, scegliendo giorno e ora in cui deve essere eseguito (per memorizzare la pianificazione si appoggia ad *Operazioni Pianificate* su *Windows* e al demone *Cron* su *Linux*).

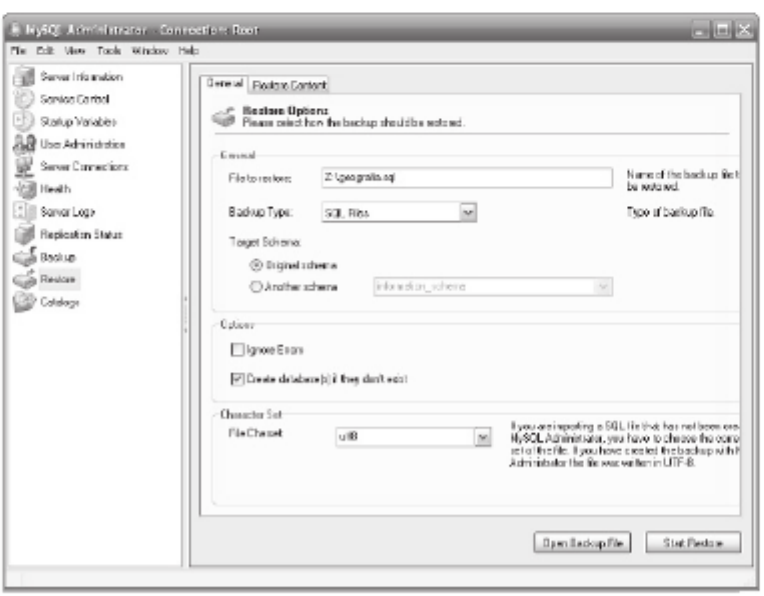


Figura 7.13
Restore

Loescher Editore - Vietata la vendita e la diffusione

Una volta creato e salvato (tramite il pulsante *Save Project*), il progetto compare nel riquadro a sinistra ed è possibile modificarne i parametri, duplicarne le impostazioni o cancellarlo.

Il file di backup è un file di testo che contiene i comandi CREATE per la creazione delle tabelle e INSERT per l'inserimento dei dati.

La sezione ***Restore*** permette di ripristinare un database dal file di backup.
Basta indicare il percorso del file di backup e alcune opzioni. Nella scheda *Restore Content* si possono specificare le singole tabelle da ripristinare.

Catalogs

La sezione ***Catalogs*** permette di visualizzare le informazioni relative agli elementi del database: tabelle contenute (con numero di record, motore di memorizzazione e dimensione), indici, viste e procedure. Da qui è possibile creare nuove tabelle o modificare quelle esistenti accedendo direttamente a *MySQL Query Browser*.

7.13 PHPMYADMIN

Introduzione

Uno degli strumenti più diffusi per interfacciarsi ad un database MySQL è **phpMyAdmin**, un programma gratuito e open source, scritto in PHP.

Per funzionare *phpMyAdmin* ha bisogno di un server Web che supporti il linguaggio PHP.
phpMyAdmin è un'applicazione PHP in esecuzione sul server; deve essere disponibile in una sottodirectory della directory radice del server Web.

Nota

Si possono modificare le impostazioni di *phpMyAdmin* mediante il file di configurazione *config.inc.php* nella directory dell'applicazione.

Per usare *phpMyAdmin* basta accedere dal browser alla directory di installazione:

```
http://localhost/phpMyAdmin
```

se *phpMyAdmin* è la sottodirectory della directory radice del server Web in cui è installato.

Nota

Per esempio se la directory radice del server Web è *C:\Programmi\Apache\htdocs* e *phpMyAdmin* viene estratto in *C:\Programmi\Apache\htdocs\phpMyAdmin*.

Accesso e pagina principale

All'accesso, *phpMyAdmin* presenta una pagina di login, dove inserire nome utente e password MySQL.

Nota

La visibilità di database e tabelle e le operazioni che si possono eseguire dipendono dai privilegi concessi con il comando GRANT all'utente con cui si effettua il login.

La pagina di *phpMyAdmin* è divisa in due frame.

Figura 7.14 *Pagina di login di phpMyAdmin*

Loescher Editore - Vietata la vendita e la diffusione

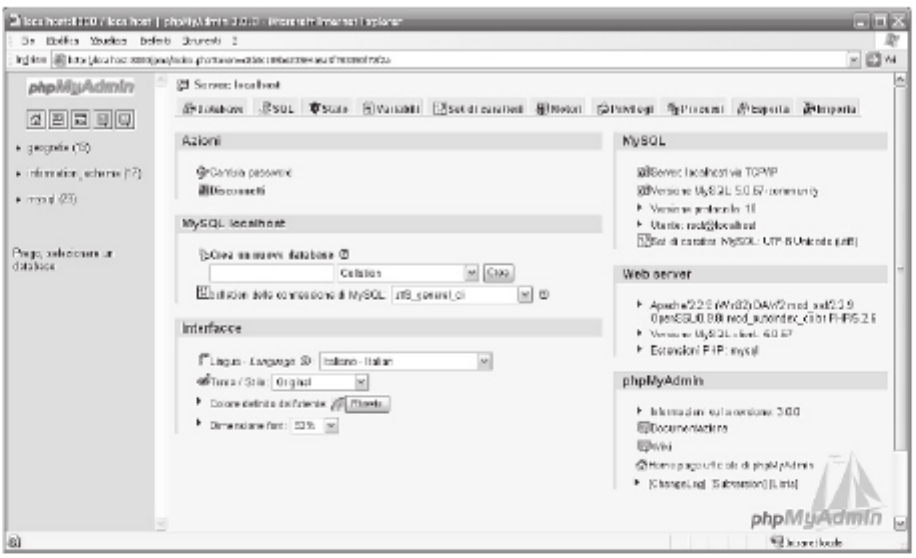


Figura 7.15
Pagina iniziale di phpMyAdmin

A sinistra è possibile **selezionare il database** desiderato.
Dopo aver selezionato un database si ottiene una lista delle tabelle del database.

Nel frame di destra è possibile creare un nuovo database; a destra vengono visualizzate alcune informazioni sulle versioni usate di MySQL, server Web e *phpMyAdmin*.
La **barra dei comandi** permette di accedere alle varie funzioni disponibili:

Database	presenta un elenco dei database e permette di gestirli;
SQL	permette di interagire con il database inserendo comandi SQL;
Stato	visualizza statistiche relative all'uso del server MySQL;
Variabili	visualizza i valori di variabili e parametri del server;
Set di caratteri	elenco dei set di caratteri;
Motori	elenco dei motori di memorizzazione;
Privilegi	consente di aggiungere nuovi utenti e assegnare i privilegi;
Processi	elenco dei processi attivi;
Esporta	permette di esportare un database (struttura e/o dati) nel formato desiderato;
Importa	permette di caricare ed eseguire un file contenente comandi SQL.

Operazioni sul database

Dopo aver selezionato un database viene visualizzata una pagina con l'**elenco delle tabelle**.

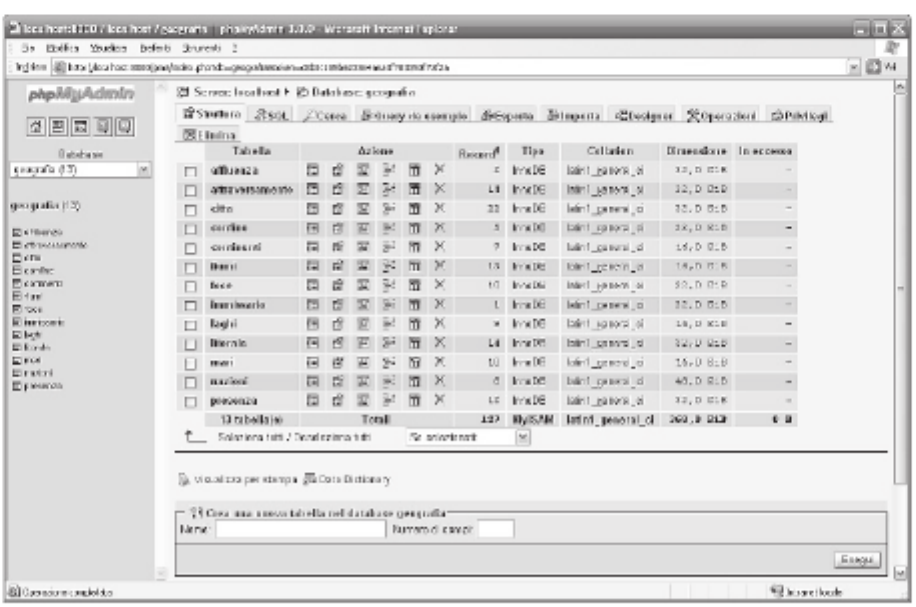


Figura 7.16
Elenco delle tabelle di un database

Loescher Editore - Vietata la vendita e la diffusione

In alto nella pagina è presentato il percorso di navigazione che riporta il nome del server e del database.

La **barra dei comandi** permette di accedere alle varie funzioni disponibili su tutto il database:

Struttura	visualizza la struttura del database (elenco delle tabelle);
SQL	permette di eseguire comandi SQL;
Cerca	permette eseguire ricerche nelle tabelle;
Query da esempio	permette di costruire query in modalità QBE;
Esporta	permette di esportare il database (struttura e/o dati) nel formato desiderato;
Importa	permette di caricare ed eseguire un file contenente comandi SQL;
Designer	visualizza in modalità grafica i collegamenti tra le relazioni;
Operazioni	permette di creare una nuova tabella e di rinominare o copiare un database;
Privilegi	consente di aggiungere nuovi utenti e assegnare i privilegi;
Elimina	cancella il database con tutte le tabelle che contiene.

Esportazione e importazione di un database

Nella finestra di esportazione è possibile scegliere le tabelle da esportare e decidere se includere anche i dati.
Tra i formati disponibili ci sono XML, PDF, CSV (Comma-separated Values) con i record uno per riga e i campi separati da virgole, Microsoft Word e Microsoft Excel, Testo, Foglio di calcolo OpenDocument (formato usato da OpenOffice) e SQL.

Nel **formato SQL** il file creato contiene i comandi SQL per la creazione delle tabelle e, se richiesto, per l'inserimento dei dati.

Il comando *Importa* accetta un file contenente comandi SQL da eseguire sul server.

La parte centrale della pagina è occupata dall'elenco delle tabelle.

In basso sono disponibili i comandi ***Visualizza per stampa*** e ***Data Dictionary***, che visualizzano in maniera sintetica la struttura delle tabelle del database.

La casella in fondo alla pagina permette di creare una nuova tabella nel database.
Dopo aver specificato il nome della tabella e il numero di campi, facendo clic sul pulsante *Esegui* si accede a una pagina in cui è possibile specificare nomi, tipi e altre caratteristiche dei campi, compresi eventuali indici.

Operazioni sulle tabelle

L'elenco delle tabelle comprende il numero di record di ciascuna, la dimensione occupata e il motore di memorizzazione utilizzato.
La colonna ***Azione*** presenta i pulsanti che permettono di compiere operazioni sulla tabella:

	Mostra: visualizza i record della tabella, che è possibile modificare o cancellare;
	Struttura: visualizza e permette di modificare la struttura della tabella;
	Cerca: permette di effettuare ricerche per valori anche approssimati nei campi della tabella;
	Inserisci: permette di inserire nuovi record nella tabella;
	Svuota: cancella tutti i record presenti, senza eliminare la tabella;
	Elimina: cancella la tabella con tutti i record contenuti.

Loescher Editore - Vietata la vendita e la diffusione

Scegliendo l'azione ***Mostra*** viene eseguita la query che seleziona tutti i campi della tabella (la query può essere modificata); è possibile decidere quante righe visualizzare per pagina e da che record partire. In questa pagina tra l'altro si può impostare un ordinamento, modificare o cancellare ciascuna riga, esportare i dati e creare una vista.

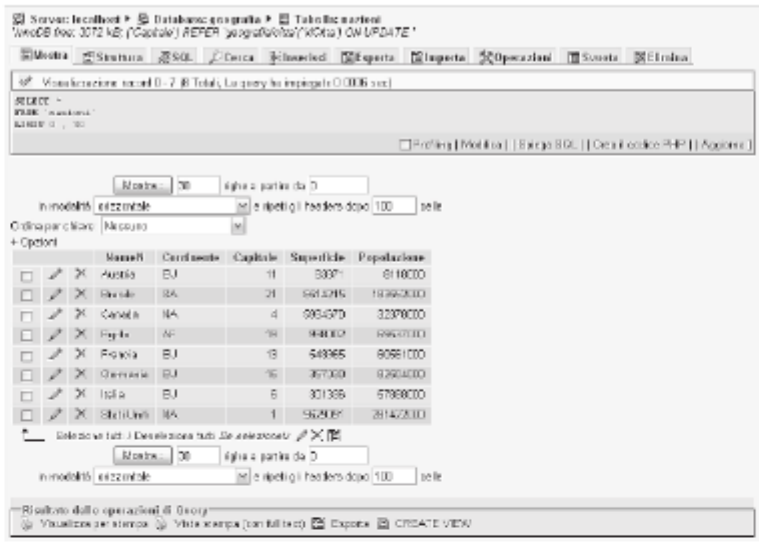


Figura 7.17
Azione Mostra su una tabella

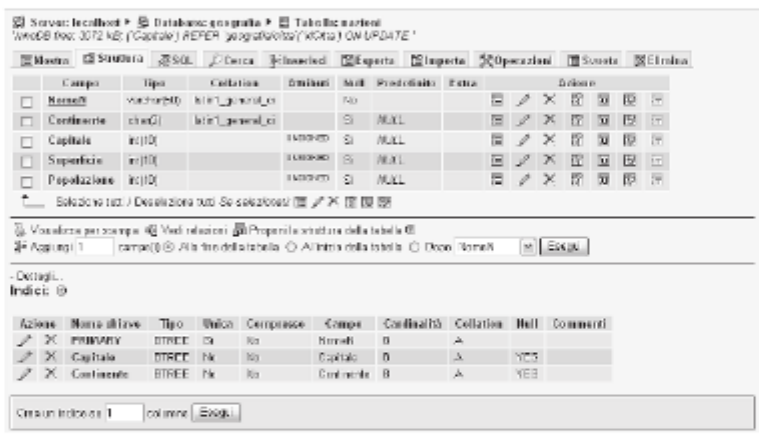


Figura 7.18
Azione Struttura su una tabella

L'azione ***Struttura*** visualizza i campi della tabella con le relative caratteristiche (nome, tipo, chiavi, indici ecc.). Facendo clic sul collegamento ***Vedi relazioni*** è possibile definire chiavi esterne e vincoli di integrità referenziale.

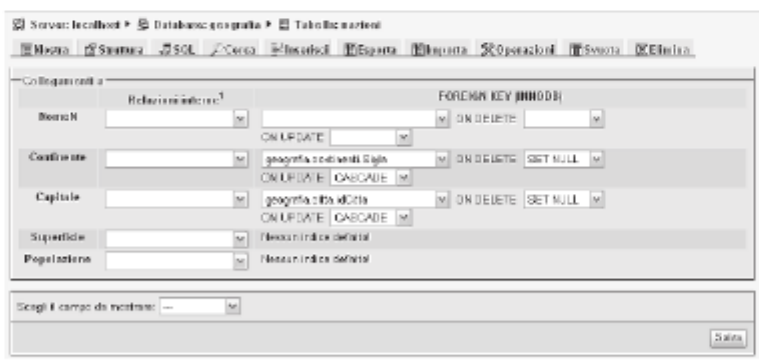


Figura 7.19
Destinazione del collegamento Vedi relazioni

Interrogazioni sul database

È possibile eseguire **query SQL** scrivendole nella pagina a cui si accede mediante il pulsante ***SQL*** nella barra dei comandi delle pagine che visualizzano database e tabelle e facendo clic sul pulsante *Esegui*.
Il risultato viene visualizzato in una nuova pagina insieme alla query, che può essere modificata facendo clic sul pulsante *Modifica*.

È possibile anche costruire query in modalità **QBE** scegliendo il pulsante ***Query da esempio*** nella barra dei comandi della pagina che elenca le tabelle del database.

Loescher Editore - Vietata la vendita e la diffusione

Nella pagina proposta è possibile selezionare i campi desiderati e impostare i **criteri di ordinamento** e uno o più **criteri di selezione** composti con gli operatori OR o AND; si possono aggiungere caselle per ulteriori campi e criteri oltre a quelle già predisposte.

La casella *Mostra* permette di stabilire se il campo deve essere visualizzato o se viene usato solo nei criteri.

In basso viene proposto l'elenco delle tabelle da cui è possibile selezionare solo quelle che interessano e il **codice** della query che si sta componendo.

Al termine è possibile eseguire la query col pulsante *Invia Query*.

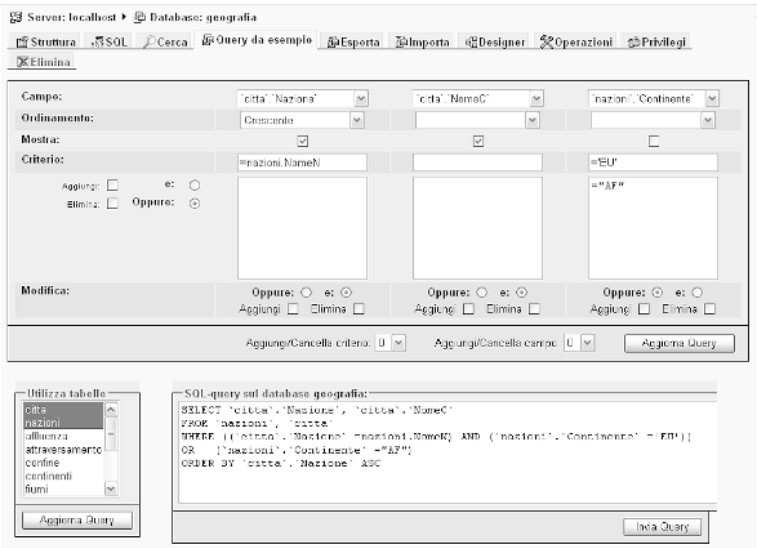


Figura 7.20 *Query da esempio*

VERIFICA

QUESTIONARIO

1. Che cosa si intende per LAMP e WAMP?
2. Che tipo di architettura ha MySQL?
3. Che cos'è un motore di memorizzazione?
4. Come si possono creare e usare variabili in MySQL?
5. Come si possono definire vincoli intrarelazionali in MySQL?
6. Quali sono le differenze tra stored routine di tipo FUNCTION e di tipo PROCEDURE?
7. Quali sono le funzioni principali di MySQL Query Browser e MySQL Administrator?
8. Quali sono i parametri necessari per la connessione all'avvio di MySQL Query Browser e MySQL Administrator?
9. Che cosa è necessario per usare phpMyAdmin e come vi si accede?

ESERCIZI

1 ..... Usando le funzioni di MySQL determinare:

- il risultato di una equazione di secondo grado, noti i coefficienti;
- un numero casuale di due cifre;
- l'età di una persona, nota la data di nascita;
- il numero di giorni trascorsi dall'inizio dell'anno scolastico e il numero di giorni che mancano alla fine;
- le iniziali del proprio nome e cognome;
- una stringa composta concatenando le prime due lettere del proprio nome, l'anno di nascita, le ultime due lettere del nome in maiuscolo;
- il numero dei giorni di un mese usando la stringa '3128313031303131303130313031'.

Loescher Editore - Vietata la vendita e la diffusione

2 ..... Dato lo schema relazionale
Studenti (Matricola, Cognome, Nome, Sesso, DataNascita, Indirizzo, Citta, Peso, Altezza, Occhiali)
dove *Matricola* è un campo contatore, *Sesso* può contenere i caratteri M o F, *Occhiali* è un valore booleano;
Materie (IdMateria, DescrizioneM)
Voti (Studente, Materia, Data, Voto)
Voti.Studente è in integrità referenziale con *Studenti.Matricola*;
Voti.Materia è in integrità referenziale con *Materie.IdMateria*;

1) creare le tabelle;
2) inserire alcuni dati;
3) creare un indice sui campi *Cognome* e *Nome* della tabella *Studenti*;
4) creare una tabella temporanea con i nomi degli studenti e il voto riportato in una specifica prova;
5) esportare in un file i risultati di tutte le prove di matematica per ogni studente con nome dello studente, data della prova e voto;
6) definire query parametriche e usarle per ricavare:
   - cognomi, nomi e indirizzi degli studenti che abitano in una data città;
   - i dati degli studenti nati prima di una certa data, ordinati in modo decrescente per data di nascita;
   - i dati degli studenti con altezza e peso superiori a un dato valore;
7) scrivere una funzione che determini la media dei voti di una prova, note la materia e la data della prova;
8) scrivere una funzione che determini la media dei voti di uno studente in una materia, noti cognome e nome dello studente e materia;
9) scrivere una procedura per inserire i risultati di una prova;
10) scrivere una procedura da usare per aggiornare periodicamente il peso degli studenti che presenti cognome, nome e peso di ogni studente e consenta di modificare il peso.

3 ..... Per il database *Geografia*, usando trigger o procedure:

1) controllare che la lunghezza di un fiume non possa essere superiore a 6671 km; se ciò si verifica:
   - impostare la lunghezza a NULL;
   - impedire l'inserimento del record;
2) controllare che non si possano inserire più di 100 città per ogni nazione;
3) registrare in una tabella apposita le variazioni delle bandiere delle nazioni, con nome della nazione, data di variazione, vecchia bandiera, nuova bandiera.

4 ..... Dato lo schema relazionale
Classi (NomeC, Aula, Rappresentante)
Studenti (Matricola, Cognome, Nome, Classe)
Classi.Rappresentante è in integrità referenziale con *Studenti.Matricola*;
usando un trigger o una procedura, controllare, all'inserimento o modifica del rappresentante di una classe, che il rappresentante appartenga effettivamente alla classe.

Dopo aver installato e avviato il server MySQL:

5 ..... Usare *MySQL Administrator* per:

- cambiare la porta di ascolto del server;
- creare un utente e assegnare solo il permesso di lettura su un database;
- consultare i file di log di sistema;
- fare il backup di un database.

Loescher Editore - Vietata la vendita e la diffusione

6 ..... Usare *MySQL Query Browser* per:
- creare tabelle;
- eseguire query;
- esportare i dati in formato XML.

7 ..... Usare *phpMyAdmin* per:
- creare un database;
- esportare il database nel formato SQL;
- eseguire query in modalità SQL e in modalità QBE.

PROPOSTE DI LAVORO

1 ..... Dato lo schema relazionale
Impiegati (Matricola, Cognome, AnnoNascita)
Reparti (IdReparto, Nome, Manager)
Lavora (Impiegato, Reparto, PercentualeTempo)
Reparti.Manager è in integrità referenziale con *Impiegati.Matricola*;
Lavora.Impiegato è in integrità referenziale con *Impiegati.Matricola*;
Lavora.Reparto è in integrità referenziale con *Reparti.IdReparto*;
deve essere possibile:
- assegnare un impiegato a un reparto controllando che la percentuale del tempo assegnato ai vari reparti non superi il 100%;
- ridurre la percentuale di tempo di lavoro di un impiegato in un reparto, eventualmente eliminando il record se la nuova percentuale è ridotta a 0;
- determinare l'elenco degli impiegati che non lavorano al 100% e la percentuale mancante.

2 ..... Dato lo schema relazionale
Conti (NumeroConto, Saldo)
Clienti (NumeroCliente, SaldoTotale, LimiteDiCredito, Nome, Indirizzo)
Titolarità (Conto, Cliente)
Operazioni (Conto, NumeroOperazione, Cliente, Tipo, Data, Ammontare)
dove *Tipo* può essere 'D' per deposito o 'P' per prelievo;
Titolarità.Conto è in integrità referenziale con *Conti.NumeroConto*;
Titolarità.Cliente è in integrità referenziale con *Clienti.NumeroCliente*;
Operazioni.Conto è in integrità referenziale con *Conti.NumeroConto*;
Operazioni.Cliente è in integrità referenziale con *Clienti.NumeroCliente*;
deve essere possibile:
- aprire un conto a un cliente;
- chiedere il saldo totale di un cliente;
- chiedere il saldo di un conto;
- effettuare un deposito su uno dei propri conti; il saldo del conto e il saldo totale di tutti i clienti titolari del conto devono essere aggiornati in tempo reale senza che si possano verificare inconsistenze dei dati;
- effettuare un prelievo da uno dei propri conti, controllando che non venga superato il limite di credito; il saldo del conto e il saldo totale di tutti i clienti titolari del conto devono essere aggiornati in tempo reale senza che si possano verificare inconsistenze dei dati;
- ottenere l'estratto conto per uno specifico conto con tutte le operazioni effettuate dai clienti del conto nel periodo tra due date;
- ottenere l'estratto conto per tutti i conti di un cliente.

Loescher Editore - Vietata la vendita e la diffusione

8 Access

Prerequisiti

- DBMS relazionali
- Linguaggio SQL

Obiettivi

Conoscenze

- Oggetti di Access (tabelle, query, maschere, report, macro e moduli)
- Modalità di visualizzazione degli oggetti
- Gestione degli eventi con macro e routine evento
- Insiemi di oggetti e riferimenti agli oggetti

Abilità/capacità

- Creare e modificare tabelle e stabilire relazioni
- Usare i fogli dati
- Creare query in modalità QBE
- Creare maschere, anche con sottomaschere
- Creare report, anche con sottoreport
- Creare ed eseguire macro
- Realizzare applicazioni

Loescher Editore - Vietata la vendita e la diffusione

8.1 IL DBMS ACCESS

Versioni di Access e formato dei file

Access è un DMBS relazionale. Le versioni di Access sono identificate dall'anno in cui sono uscite; la versione più recente è Access 2007.

Nota Nel seguito del testo si farà riferimento ad Access 2007.

Le versioni di Access fino ad Access 2003 memorizzano il database in file con estensione ***.mdb*** (acronimo di Microsoft DataBase); il formato *.mdb* ha subito modifiche nel corso delle versioni rendendo non compatibili database creati con versioni diverse.

Access 2007 introduce il nuovo formato di database ***.accdb*** (acronimo di ACCess DataBase) non compatibile con il formato *.mdb*.

Access 2007 è in grado di gestire correttamente anche il formato *.mdb* di Access 2002 e 2003; per l'apertura di database creati con versioni antecedenti la 2002 chiede se si desidera effettuare la conversione al formato *.accdb*.

Gli oggetti di Access

Tutti gli elementi di Access sono oggetti:

tabelle	corrispondono alle relazioni del modello logico; ogni riga rappresenta un record, ogni colonna rappresenta un campo;
query	operazioni sui dati; le query di selezione permettono di effettuare interrogazioni sui dati, le query di comando permettono di effettuare operazioni di aggiornamento;
maschere	permettono di definire videate per immettere dati in una tabella, ma anche di creare finestre di dialogo o pannelli di comandi;
report	permettono di predisporre opportunamente le informazioni per la stampa;
macro	sono sequenze di comandi raggruppati in un'unica operazione che permettono di automatizzare lavori ripetitivi;
moduli	sono segmenti di codice Visual Basic.

Gli oggetti e i loro elementi (come i campi delle tabelle) posseggono delle proprietà che vengono utilizzate per impostare le caratteristiche dell'oggetto o dell'elemento.

Interfaccia di Access 2007

Le finestre di Access 2007 hanno un pulsante ***Office*** per accedere ai menu principali e barre degli strumenti legate a uno specifico contesto/menu. Ciascuna barra è suddivisa in gruppi di strumenti affini.

Nota Il layout di Access 2007 è analogo a quello delle finestre del sistema operativo *Windows Vista*.

Figura 8.1 *Pulsante Office con la barra di accesso rapido e barra degli strumenti Home*

Loescher Editore - Vietata la vendita e la diffusione

Apertura o creazione di un database

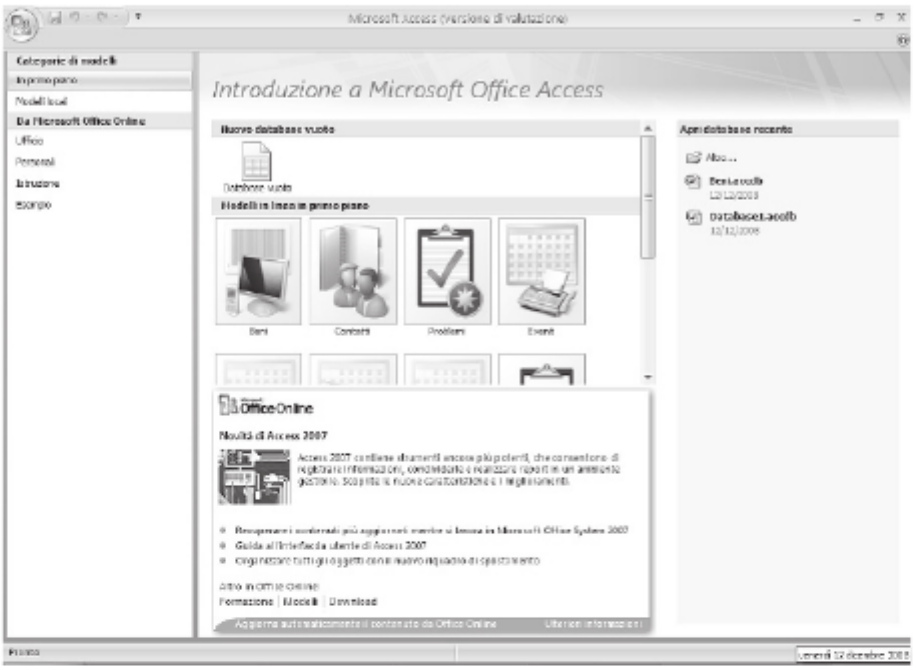


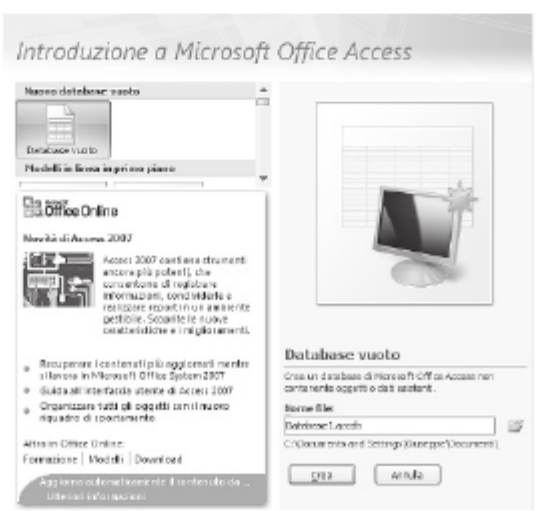
Figura 8.2 *Finestra di avvio di Access 2007 (se il computer è collegato a Internet la finestra di avvio presenta di volta in volta le novità disponibili)*

La prima cosa da fare all'avvio è aprire un database o crearne uno nuovo.

Si può **aprire un database** dalla lista dei database creati o aperti recentemente o facendo clic sul pulsante *Office* e scegliendo *Apri*; il database scelto viene aperto in una finestra che riporta sulla barra del titolo il nome del database e la versione di Access con cui è stato creato e che presenta la barra degli strumenti *Home* e la finestra *Tabelle* con le tabelle del database.

Nota In base alle impostazioni può comparire un avviso di protezione nella *Barra messaggi*.

Nota Quando si apre un database, viene creato nella stessa cartella un file temporaneo, eliminato automaticamente alla chiusura del database, che può impedire ad altri utenti di modificare il database per preservare l'integrità dei dati; il file ha estensione *.laccdb* per i database di formato *.accdb* e *.ldb* per i database di formato *.mdb*.



Per **creare un database** bisogna specificarne il nome e la posizione.
Si può creare un database da un modello predefinito scegliendolo tra quelli proposti nella scheda *Modelli in linea in primo piano* (è necessaria la connessione a Internet) oppure tra quelli della voce *Modelli locali* della scheda *Categorie di Modelli*.
Per creare un database partendo da zero bisogna selezionare ***Database vuoto*** nella scheda *Nuovo database vuoto*.

Figura 8.3 *Creazione di un nuovo database*

Loescher Editore - Vietata la vendita e la diffusione

Si apre la finestra *Strumenti Tabella* per l'immissione dei dati; se si è scelto un modello vengono presentati campi predefiniti.

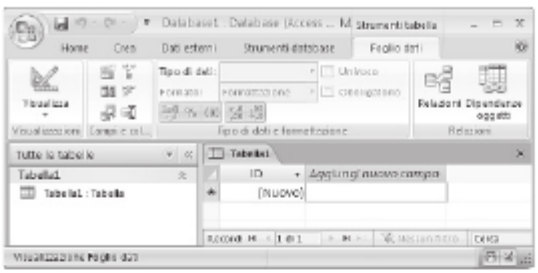


Figura 8.4
La finestra Strumenti Tabella

Una volta creato o modificato il database si può salvarlo scegliendo ***Salva*** o ***Salva con nome*** dal pulsante *Office* e selezionando il formato desiderato.

Backup di un database

Per effettuare un **backup** del database basta fare una copia del file (*.accdb* o *.mdb*). Dal pulsante *Office* si può scegliere *Gestisci* e poi *Backup database* (apre la finestra *Salva con nome*).

Per **ripristinare** un database basta ricopiare il file di backup nella directory desiderata.

Esportazione di dati

Si possono **esportare i dati** di un database con gli strumenti del gruppo *Esporta* della barra *Dati esterni*. Tra i formati disponibili ci sono Excel, RTF e testo ma anche XML, HTML e altri formati di database.

Documentazione del database

È possibile visualizzare o stampare la definizione di un oggetto del database (struttura, proprietà ecc.) mediante il gruppo ***Analizza*** della barra *Strumenti database*.

Lo strumento *Analizzatore database* apre la finestra *Analizzatore* in cui è possibile scegliere la scheda relativa al tipo di oggetti (tabelle, query, maschere, report ecc.) e gli oggetti specifici.
Si può ottenere la documentazione di tutti gli oggetti del database selezionando tutti gli oggetti con il pulsante *Seleziona tutto* della scheda *Tutti i tipi di oggetti*.
La definizione degli oggetti selezionati viene visualizzata in anteprima di stampa e può essere stampata con lo strumento *Stampa* dalla barra *Anteprima di stampa*.

Nota
Lo strumento *Analizza prestazioni* suggerisce modifiche per incrementare l'efficienza del database. Lo strumento *Analizza tabella* avvia una procedura guidata per l'eventuale suddivisione di una tabella in più tabelle.

Creazione di oggetti

Gli oggetti del database possono essere creati usando la barra degli strumenti ***Crea*** che contiene un gruppo di strumenti per ogni tipo di oggetti.

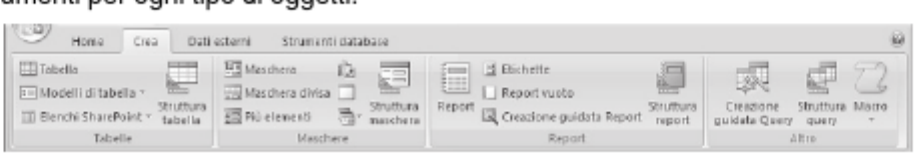


Figura 8.5
La barra degli strumenti Crea

Loescher Editore - Vietata la vendita e la diffusione

Per **creare un oggetto** basta scegliere uno degli strumenti del gruppo relativo.

Dopo la scelta dello strumento di creazione la barra *Crea* viene sostituita da una barra degli strumenti più adatta al contesto: per esempio la scelta dello strumento *Tabella* presenta la barra *Foglio dati*, mentre la scelta di *Struttura tabella* presenta la barra *Progettazione*.

Per **salvare** un oggetto si può usare lo strumento *Salva* della barra di accesso rapido (in genere presente accanto al pulsante *Office*) o il comando *Salva* del menu di scelta rapida che si ottiene facendo clic destro sulla scheda dell'oggetto. Dopo il salvataggio il nome dell'oggetto compare sulla linguetta (tab) della scheda.

Per la creazione di query, maschere e report sono disponibili le **procedure guidate** *Creazione guidata Query*, *Creazione guidata Maschera* e *Creazione guidata Report* che permettono di creare gli oggetti impostando automaticamente le caratteristiche desiderate.

Regole per la scelta degli identificatori

Gli **identificatori** (nomi di oggetti, campi ecc.) possono essere lunghi fino a 64 caratteri e possono essere formati da qualsiasi combinazione di lettere, numeri, spazi e caratteri speciali, ad eccezione del punto, del punto esclamativo, dell'accento grave e delle parentesi quadre (caratteri . ! ‘ [ ]).

Nota È opportuno usare nomi diversi da quelli predefiniti e non usare nomi che contengono spazi soprattutto se devono essere utilizzati in codice Visual Basic.

Tutti gli oggetti creati vengono elencati nel ***Riquadro di spostamento***.

Nel riquadro gli oggetti possono essere raggruppati per tipo (allora il riquadro ha il nome *Tutti gli oggetti di Access*) oppure in base alla loro relazione con le tabelle (allora il riquadro prende il nome *Tutte le tabelle*).

I comandi principali relativi agli oggetti sono disponibili dal menu di scelta rapida che si ottiene facendo clic destro sull'oggetto nel riquadro di spostamento.

Per fare una copia di un oggetto si usa il comando *Salva con nome*, *Salva oggetto con nome* del pulsante *Office*.

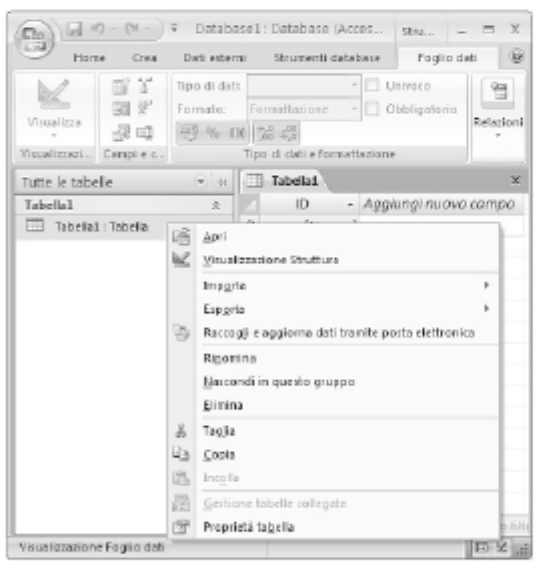


Figura 8.6
Riquadro di spostamento e menu di scelta rapida relativo alla tabella

Nelle versioni di Access fino alla 2003 gli oggetti e le operazioni principali sono gestibili dalla **finestra del database** presentata alla creazione o apertura di un database.

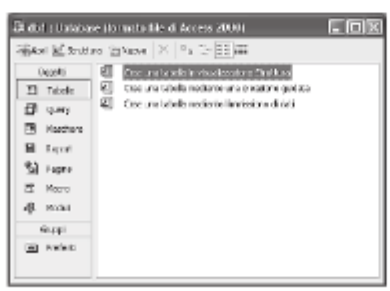


Figura 8.7
Finestra del database

Loescher Editore - Vietata la vendita e la diffusione

8.2 TABELLE

Creazione di tabelle

Si possono **creare** le tabelle mediante gli strumenti del gruppo ***Tabelle*** della barra *Crea*:

Tabella	presenta una scheda suddivisa in celle in cui inserire direttamente i dati (senza doverne definire il tipo); vengono creati nuovi record o nuovi campi quando si inseriscono dati rispettivamente in una nuova riga o una nuova colonna;
Modelli di tabella	permette di scegliere un modello in base al quale crea la struttura della tabella e presenta una scheda per l'inserimento dei dati;
Struttura tabella	presenta una scheda per la definizione dei campi: nome, tipo e proprietà di ciascun campo.

Per **salvare** la tabella si può usare lo strumento *Salva* della barra di accesso rapido o il comando *Salva* del menu di scelta rapida che si ottiene facendo clic destro sulla scheda della tabella. Dopo il salvataggio il nome della tabella compare sulla linguetta della scheda.

Nota

Se la tabella non contiene una chiave compare una finestra di dialogo che chiede se si desidera crearla; rispondendo di sì viene creato automaticamente un campo di tipo *Contatore*.

Modifica di tabelle

Si può **modificare** una tabella aprendola dal riquadro di spostamento in modalità ***Visualizzazione Struttura***; è possibile per esempio cambiare il tipo di dati di un campo o inserire o eliminare campi.

Attenzione

Se la tabella contiene già dei dati, variare il tipo di dati di un campo può non essere possibile (per esempio nessun tipo di dati può essere convertito nel tipo *Contatore*) o può comportare la perdita di dati.

Se sono stati definiti altri oggetti basati sulla tabella (come maschere, query o report) modificare la tabella può richiedere la modifica degli altri oggetti.

Proprietà di tabella

Lo strumento ***Finestra delle proprietà*** del gruppo *Mostra/Nascondi* della barra *Progettazione* (attiva quando la tabella è in visualizzazione struttura) o della barra *Strumenti database* apre la finestra delle proprietà della tabella.

Alcune proprietà sono:

Descrizione	testo di descrizione associato alla tabella;
Valido se	permette di specificare un criterio che l'intero record deve soddisfare (per esempio confronti tra campi del record);
Messaggio errore	testo visualizzato se il record non soddisfa il criterio specificato nella proprietà *Valido se*;
Filtro	permette di specificare un criterio per selezionare i record della tabella;
Ordina per	permette di indicare il nome di uno o più campi in base ai quali ordinare la tabella (in senso crescente);

Loescher Editore - Vietata la vendita e la diffusione

Definizione dei campi

In visualizzazione struttura la scheda relativa alla tabella presenta nella parte superiore lo schema per la definizione dei campi; ogni campo deve essere definito in una riga.
Lo schema contiene le colonne:

Nome campo	nome del campo;
Tipo dati	tipo di dati, scelto da una casella combinata; il valore di default è il tipo *Testo*;
Descrizione	facoltativa, compare sulla riga di stato quando ci si posiziona sul campo in fase di visualizzazione o aggiornamento dei dati.

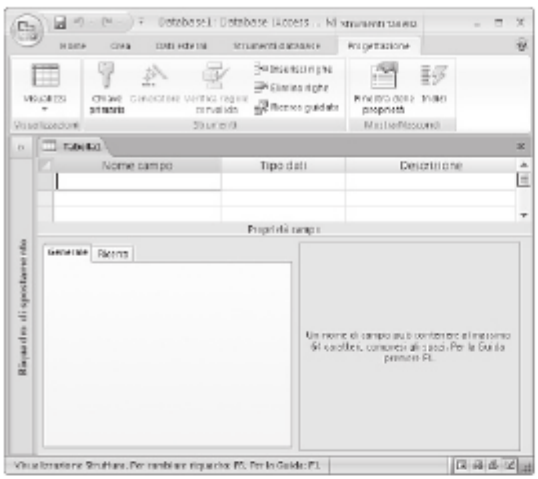


Figura 8.8 *Visualizzazione struttura*

Si può inserire un nuovo campo o cancellare un campo con gli strumenti *Inserisci righe* ed *Elimina righe* del gruppo *Strumenti* della barra *Progettazione*
Si può spostare un campo selezionandolo e trascinandolo nella nuova posizione (si può selezionare un campo facendo clic nella casella a sinistra, chiamata selettore di campo).

Nel riquadro ***Proprietà campo*** la scheda *Generale* permette di impostare le proprietà che caratterizzano ciascun campo.
Le proprietà visualizzate si riferiscono al campo selezionato nella parte superiore.

> Per alcune proprietà è possibile selezionare un valore da un elenco o usare un generatore che facilita l'impostazione del valore.
> **Nota**

Tipi di dati

Testo	caratteri alfanumerici (fino a 255); la lunghezza del campo può essere impostata con la proprietà *Dimensione campo*;
Memo	testo lungo (fino a 65536 caratteri);
Numerico	valori numerici; il tipo di valore numerico (*Byte*, *Intero*, *Intero lungo*, *Precisione semplice*, *Precisione doppia*, *Decimale* e *ID Replica*) può essere precisato con la proprietà *Dimensione campo*;
Data/ora	date o orari; il formato della data (data generica, estesa, breve e in cifre) o dell'ora (ora estesa, breve 12h e breve 24h) può essere precisato con la proprietà *Formato*;
Valuta	valori monetari e numeri con 4 cifre a destra e fino 15 cifre a sinistra del separatore decimale;

> Per un valore in euro conviene impostare la proprietà *Formato* al valore *euro* e la proprietà *Posizioni decimali* al valore 2.

Contatore	numero progressivo o valore casuale secondo l'impostazione della proprietà *Nuovi valori*; occupa 4 byte (una chiave esterna associata a un campo di tipo *Contatore* deve essere di tipo numerico *Intero lungo*);
Sì/No	valore booleano (del tipo vero/falso);
Oggetto OLE	oggetto realizzato con un'altra applicazione (foglio di calcolo, immagine ecc.);

> *Oggetto OLE* è mantenuto per compatibilità con le versioni precedenti, è sostituito in Access 2007 dal più efficiente tipo *Allegato*.
> **Nota**

Loescher Editore - Vietata la vendita e la diffusione

Collegamento ipertestuale
testo che rappresenta un collegamento ipertestuale; è formato da tre parti ciascuna lunga fino a 2048 caratteri: il testo visualizzato come collegamento, l'URL del file e la descrizione da visualizzare quando il mouse è sopra al collegamento;

Allegato qualsiasi tipo di file;

Ricerca guidata... crea un campo che consente di scegliere un valore di un'altra tabella o di un elenco di valori utilizzando una casella combinata.

Proprietà dei campi

Il tipo di proprietà definibili dipende dal tipo di dati del campo.

Alcune proprietà della scheda ***Generale*** sono:

Dimensione campo per i campi di tipo *Testo* un numero da 1 a 255 che indica il numero massimo di caratteri (viene utilizzato solo lo spazio di memoria necessario per i caratteri effettivi); per i campi di tipo *Numerico* il tipo di valore numerico: *Byte* (1 byte), *Intero* (2 byte), *Intero lungo* (4 byte), *Precisione semplice* (4 byte), *Precisione doppia* (8 byte), *Decimale* e *ID Replica*;

Formato influenza l'aspetto dei dati; le impostazioni possibili dipendono dal tipo di dato; si possono definire anche dei formati personalizzati;

Posizioni decimali numero di cifre decimali da visualizzare per i campi di tipo *Numerico* e *Valuta*.

Maschera di input supportata solo dai campi di tipo *Testo* e *Data/ora*, permette di definire un formato da utilizzare per l'immissione dei dati; è possibile per esempio controllare l'inserimento di caratteri del tipo corretto in ogni posizione, visualizzare automaticamente caratteri alfanumerici uguali per tutti i campi, nascondere informazioni riservate (come le password);

Etichetta testo utilizzato al posto del nome del campo durante la visualizzazione di oggetti come fogli dati, maschere o report (diventa per esempio l'intestazione di colonna per il campo in visualizzazione foglio dati);

Valore predefinito valore inserito automaticamente nel campo per ogni nuovo record creato;

Valido se permette di impostare un criterio che i valori del campo devono soddisfare per essere ritenuti validi;

Messaggio errore testo visualizzato se il valore inserito nel campo non soddisfa il criterio specificato nella proprietà *Valido se*;

Richiesto indica se è necessario inserire un valore nel campo (distingue tra campi obbligatori o facoltativi);

Consenti lunghezza zero
proprietà dei campi di tipo *Testo* e *Memo*; indica se è possibile immettere stringhe vuote ("");

Nuovi valori proprietà dei campi di tipo *Contatore*; distingue tra numeri progressivi (impostazione *Incremento*) o casuali (impostazione *Casuale*);

Indicizzato permette di creare un indice sul campo (valori *No*, *Sì (Duplicati ammessi)*, *Sì (Duplicati non ammessi)*).

Esempio 8.1

Creazione della tabella:
Nazioni(*NomeN*, *Continente*, *Capitale*, *Superficie*, *Popolazione*, *Bandiera*)

Nazioni

Nome campo	Tipo dati	Descrizione
NomeN	Testo	
Continente	Testo	
Capitale	Numerico	
Superficie	Numerico	
Popolazione	Numerico	
Bandiera	Allegato	

Loescher Editore - Vietata la vendita e la diffusione

NomeN Testo

Generale | Ricerca

Dimensione campo	20
Formato	
Maschera di input	
Etichetta	
Valore predefinito	
Valido se	
Messaggio errore	
Richiesto	Sì
Consenti lunghezza zero	No
Indicizzato	No
Compressione Unicode	Sì
Modalità IME	Nessun controllo
Modalità frase IME	Nessuna conversione
Smart tag	

Continente Testo

Generale | Ricerca

Dimensione campo	20
Formato	
Maschera di input	
Etichetta	
Valore predefinito	
Valido se	
Messaggio errore	
Richiesto	Sì
Consenti lunghezza zero	No
Indicizzato	No
Compressione Unicode	Sì
Modalità IME	Nessun controllo
Modalità frase IME	Nessuna conversione
Smart tag	

Capitale Numerico

Generale | Ricerca

Dimensione campo	Intero lungo
Formato	
Posizioni decimali	Automatiche
Maschera di input	
Etichetta	
Valore predefinito	
Valido se	
Messaggio errore	
Richiesto	No
Indicizzato	No
Smart tag	
Allineamento testo	Standard

Superficie Numerico

Generale | Ricerca

Dimensione campo	Intero lungo
Formato	Standard
Posizioni decimali	0
Maschera di input	
Etichetta	
Valore predefinito	Null
Valido se	>0
Messaggio errore	La superficie deve essere un nur
Richiesto	No
Indicizzato	No
Smart tag	
Allineamento testo	Standard

Popolazione Numerico

Generale | Ricerca

Dimensione campo	Intero lungo
Formato	Standard
Posizioni decimali	0
Maschera di input	
Etichetta	
Valore predefinito	Null
Valido se	>0
Messaggio errore	La popolazione deve essere un i
Richiesto	No
Indicizzato	No
Smart tag	
Allineamento testo	Standard

Bandiera Allegato

Generale | Ricerca

Etichetta	
Richiesto	No

Il campo *Continente* è indicizzato.
Il campo *Capitale* è di tipo *Numerico* dimensione *Intero lungo* perché sarà una chiave esterna.
Il campo *Bandiera* è di tipo *Allegato* per contenere un'immagine.

Nota

Se la versione di Access utilizzata non è la 2007 il tipo *Allegato* non è disponibile; può essere sostituito con il tipo *Oggetto OLE*.

Per il momento non è impostata alcuna chiave; salvando la tabella si può rispondere di no alla richiesta di creazione di una chiave primaria.

Loescher Editore - Vietata la vendita e la diffusione

Definizione della chiave

Per definire la chiave primaria di una tabella basta selezionare il campo desiderato in visualizzazione struttura della tabella e attivare lo strumento ***Chiave primaria*** del gruppo *Strumenti* della barra *Progettazione*; sul selettore di campo compare un simbolo di chiave.

Nota Per rimuovere la chiave basta selezionare il campo nella griglia di definizione della tabella e disattivare lo strumento *Chiave primaria*.

Per definire una chiave primaria multicampo basta selezionare i campi nella griglia di definizione della tabella (tenendo premuto *Ctrl*) e attivare lo strumento *Chiave primaria*.

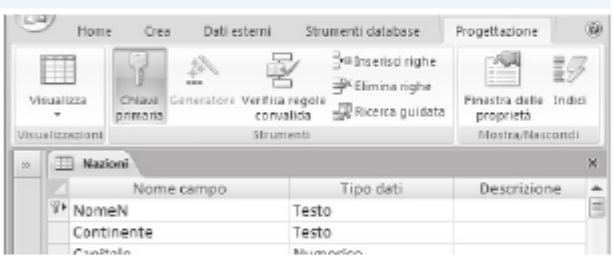
Nota L'ordine dei campi nella chiave dipende dalla loro posizione nella struttura della tabella.

Campo chiave

Per la chiave primaria viene automaticamente creato un **indice** e i record vengono visualizzati secondo l'ordine definito dalla chiave.
Quando si inserisce un record in una tabella che ha una chiave primaria viene controllato che il valore della chiave non sia *null* e non che sia già presente un record con la stessa chiave.

Esempio 8.2

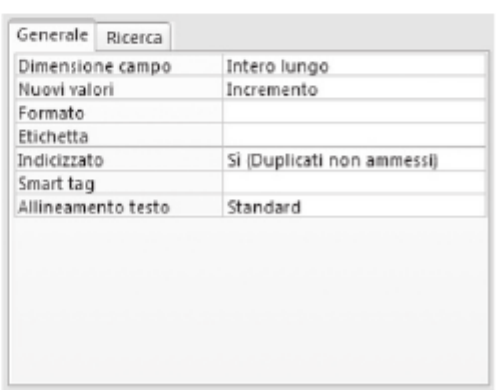
Modifica della tabella *Nazioni* per far diventare chiave il campo *NomeN*.



Esempio 8.3

Creazione della tabella:
Città(IdCitta, NomeC, Nazione, Popolazione)
Il campo *IdCitta*, di tipo *Contatore*, è la chiave.
Il campo *Nazione* è indicizzato.

IdCitta Contatore



NomeC Testo

Loescher Editore - Vietata la vendita e la diffusione

Nazione Testo

Generale	Ricerca
Dimensione campo	20
Formato	
Maschera di input	
Etichetta	
Valore predefinito	
Valido se	
Messaggio errore	
Richiesto	No
Consenti lunghezza zero	Sì
Indicizzato	Sì (Duplicati ammessi)
Compressione Unicode	Sì
Modalità IME	Nessun controllo
Modalità frase IME	Nessuna conversione
Smart tag	

Popolazione Numerico

Generale	Ricerca
Dimensione campo	Intero lungo
Formato	Standard
Posizioni decimali	0
Maschera di input	
Etichetta	
Valore predefinito	
Valido se	
Messaggio errore	
Richiesto	No
Indicizzato	No
Smart tag	
Allineamento testo	Standard

Indici e chiavi multicampo

Gli **indici** su un solo campo possono essere definiti impostando la proprietà ***Indicizzato*** dei campi della tabella.
Per definire **indici multicampo** è necessario usare la finestra ***Indici*** a cui si accede scegliendo lo strumento *Indici* del gruppo *Mostra/Nascondi* della barra *Progettazione*.
Nella colonna *Nome campo* vanno indicati, uno per riga, i nomi dei campi dell'indice, che possono essere selezionati dalla casella combinata che appare facendo clic sulla colonna; nella colonna *Nome indice* va indicato il nome dell'indice soltanto sulla prima riga, lasciando vuota la cella nelle righe successive.
Si possono cancellare gli indici rimuovendo le righe corrispondenti.

Nota

L'indice viene creato quando si salva la struttura della tabella e viene aggiornato man mano che si inseriscono o modificano i record.

La finestra *Indici* riporta anche l'indice sul campo chiave con nome *PrimaryKey*, il simbolo di chiave a sinistra e le proprietà *Primario* e *Univoco* impostate a *Sì*.
Cancellando la riga vengono eliminati sia la chiave che l'indice; per eliminare la chiave conservando l'indice basta impostare la proprietà *Primario* al valore *No*.

Usando la finestra *Indici* è possibile creare una **chiave multicampo** ordinando i campi della chiave anche in modo diverso dall'ordine in cui sono definiti nella struttura della tabella; basta definire un indice multicampo e impostare la proprietà *Primario* a *Sì*.

Esempio 8.4

Aggiunta alla tabella *Nazioni* dell'indice multicampo per i campi *Continente* e *NomeN*; la finestra riporta tutti gli indici.

Indici: Nazioni

Nome indice	Nome campo	Criterio ordinamento
PrimaryKey	NomeN	Crescente
Continente	Continente	Crescente
ContinenteNazione	Continente	Crescente
	NomeN	Crescente

Proprietà indice

Primario	Sì
Univoco	Sì
Ignora Null	No

Nome dell'indice. Ciascun indice può utilizzare al massimo 10 campi.

Loescher Editore - Vietata la vendita e la diffusione

Il foglio dati

Per inserire, visualizzare o modificare i **dati**, bisogna aprire la tabella in modalità ***Visualizzazione Foglio dati***.

Nota: Si può aprire una tabella in modalità foglio dati scegliendo *Apri* dal menu di scelta rapida del riquadro di spostamento.

Si può passare dalla visualizzazione struttura alla visualizzazione foglio dati e viceversa con lo strumento ***Visualizza*** (gruppo *Visualizzazioni*) delle barre *Progettazione* o *Home*.

Nota: La modalità di visualizzazione può essere scelta anche dal menu di scelta rapida che si ottiene facendo clic destro sulla scheda della tabella.

Se la tabella è vuota il foglio dati presenta una tabella composta di due sole righe, una con i nomi dei campi della tabella, l'altra vuota, per inserire il primo record.
Se la tabella contiene dati, il foglio dati presenta una riga per ogni record, più una riga vuota per inserire un nuovo record.

La casella a sinistra è chiamata **selettore di record** (facendo clic sulla casella si seleziona il record corrispondente; il record corrente è evidenziato da un colore diverso); nel selettore del record vuoto è presente un asterisco.

Nota: Nelle versioni di Access fino alla 2003 il record corrente è segnalato da un triangolo nel selettore di record.

Ogni casella della prima riga, con il nome del campo, costituisce il **selettore di campo** e permette di selezionare l'intera colonna.

Per **inserire un record** basta scrivere il valore dei campi nella riga vuota; quando si digita il primo carattere viene presentata una nuova riga vuota.

Per **modificare un record** basta posizionare il cursore sul campo desiderato e digitare il nuovo valore (ci si può spostare usando i tasti ↑ ↓ → ←, il tasto di tabulazione e il tasto *Invio*).

Nota: Si può annullare una modifica con lo strumento *Annulla* della barra di accesso rapido.

Quando si crea o si modifica un record, nel selettore di record compare una matita.
Il record viene **salvato** quando si passa ad un altro record.

Per **selezionare** un record basta fare clic sul selettore del record; per selezionare più record bisogna fare clic sul selettore del primo record e trascinare il puntatore fino al selettore dell'ultimo record da includere; per selezionare record non contigui basta tenere premuto il tasto *Ctrl* mentre si selezionano; per selezionare tutti i record del foglio dati si può fare clic sul selettore nell'angolo superiore sinistro del foglio dati.

Per **posizionarsi** su un record visibile basta fare clic sulla riga corrispondente.
Per posizionarsi su un record qualsiasi si possono usare i pulsanti di spostamento posti nella barra di stato della scheda relativa alla tabella; per passare ad un record specifico basta digitare il numero del record nell'apposita casella.

Per operare sui record si possono usare gli strumenti della barra ***Home***:

gruppo ***Record***
- *Salva* — salva un record;
- *Elimina* — cancella un record;

gruppo ***Trova***
- *Seleziona* — seleziona un record;

Loescher Editore - Vietata la vendita e la diffusione

Seleziona tutto seleziona tutti i record;
Vai a passa a un altro record (al primo, all'ultimo, al precedente, al successivo o al nuovo);
gruppo ***Appunti***
Taglia, *Copia* e *Incolla* per spostare o copiare record.

Nota
Quando si incollano record in un foglio dati diverso i dati vengono incollati nelle colonne in base all'ordine dei campi indipendentemente dai nomi.

Stampa di un foglio dati

Si può **stampare** il foglio dati col comando *Stampa* del pulsante *Office*. Si possono stampare solo alcuni record del foglio selezionandoli prima di richiamare il comando di stampa e scegliendo il pulsante di opzione *Record selezionato/i* nella sezione *Intervallo stampa* della finestra *Stampa*.

Il **layout** del foglio dati può essere modificato in vari modi agendo direttamente sul foglio dati o mediante lo strumento *Altro* del gruppo *Record* della barra *Home*.

Per spostare una colonna basta selezionarla (facendo clic sul selettore di campo quando il puntatore assume la forma di una freccia rivolta verso il basso) e trascinarla nella posizione desiderata.

Per modificare la larghezza di una colonna basta trascinare a destra o a sinistra il bordo del selettore di campo (il puntatore assume la forma di una freccia a due punte); si può fare in modo che si adatti automaticamente al valore più largo facendo doppio clic sul bordo destro del selettore di campo.

Per modificare l'altezza di una riga basta trascinare in alto o in basso il bordo del selettore di record (il puntatore assume la forma di una freccia a due punte).

Con lo strumento *Altro*:

Larghezza colonne...	modifica la larghezza delle colonne; il pulsante *Adatta* adatta automaticamente la larghezza al valore più largo;
Altezza righe...	modifica l'altezza delle righe;
Nascondi colonne	permette di nascondere una o più colonne;
Scopri colonne...	permette di scegliere le colonne da visualizzare, selezionandole;
Blocca	permette di bloccare una o più colonne; le colonne bloccate rimangono sempre visualizzate anche quando si scorre la finestra verso destra se le colonne sono troppe per essere visualizzate tutte contemporaneamente; *Sblocca* annulla l'impostazione.

Gli strumenti del gruppo *Carattere* della barra *Home* permettono di modificare il tipo di carattere, la dimensione, lo stile, l'allineamento, i bordi, i colori del testo e dello sfondo utilizzati nel foglio dati.

Esempio 8.5

Inserimento di dati nella tabella *Nazioni* dell'esempio 8.1.

Nazioni

NomeN	Continente	Capitale	Superficie	Popolazione	
Austria	EU	11	83.871	8.118.000	(1)
Brasile	SA	21	8.514.215	183.552.000	(1)
Canada	NA	4	9.984.670	32.378.000	(1)
Egitto	AF	19	998.002	69.537.000	(1)
Francia	EU	13	543.965	60.561.000	(1)
Germania	EU	16	357.030	82.504.000	(1)
Italia	EU	6	301.336	57.888.000	(1)
Stati Uniti	NA	1	9.629.091	281.422.000	(1)
					(0)

Record: 4 di 9 Nessun filtro Cerca

Loescher Editore - Vietata la vendita e la diffusione

Elenchi di valori

È possibile effettuare l'inserimento dei valori in campo scegliendoli da un elenco prestabilto, presentato mediante una casella di riepilogo o una casella combinata, predisponendo i valori nella scheda ***Ricerca*** delle proprietà del campo.

L'**elenco dei valori** può essere definito:

- al momento, impostando *Tipo origine riga* a *Elenco valori* e scrivendo i valori separati dal simbolo punto e virgola nella casella *Origine riga* (è disponibile anche un generatore per facilitare la scrittura dei valori);
- prendendo i valori da un campo di una tabella impostando *Tipo origine riga* a *Tabella/query* e *Origine riga* al nome della tabella; la proprietà *Colonna associata* indica il numero d'ordine della colonna nella tabella.

Esempio 8.6

Si possono modificare le impostazioni del campo *Continente* della tabella *Nazioni* in modo che il valore possa essere scelto da un elenco prestabilito.

L'elenco può essere definito nella scheda *Ricerca*, eventualmente usando il generatore a cui si può accedere con il pulsante con i tre puntini che compare quando si fa clic sulla casella *Origine riga*.

Generale | Ricerca

Visualizza controllo	Casella combinata
Tipo origine riga	Elenco valori
Origine riga	"AF";"AS";"CA";"EU";"NA";"OC";"SA"
Colonna associata	1
Numero colonne	1
Intestazioni colonne	No
Larghezza colonne	
Righe in elenco	16
Larghezza elenco	Automatica
Solo in elenco	Sì
Consenti valori multipli	No
Consenti modifiche a elenco valori	No
Maschera di modifica voci di elenco	
Mostra solo valori origine riga	No

Una soluzione migliore consiste nel definire una tabella *Continenti* con sigla e nome del continente per esteso e impostare la proprietà *Origine riga* al nome della tabella.

Continenti

Nome campo	Tipo dati
Sigla	Testo
NomeCon	Testo

Generale | Ricerca

Visualizza controllo	Casella combinata
Tipo origine riga	Tabella/query
Origine riga	Continenti
Colonna associata	1
Numero colonne	1
Intestazioni colonne	No
Larghezza colonne	
Righe in elenco	16
Larghezza elenco	Automatica
Solo in elenco	No
Consenti valori multipli	No
Consenti modifiche a elenco valori	No
Maschera di modifica voci di elenco	
Mostra solo valori origine riga	No

In entrambi i casi nel foglio dati è possibile scegliere il continente da una casella combinata.

Nazioni

NomeN	Continente	Capitale	Superficie	Popolazione
Austria	EU	11	83.871	8.118.000
Brasile	AF	21	8.514.215	183.552.000
Canada	AS	4	9.984.670	32.378.000
Egitto	CA	19	998.002	69.537.000
Francia	EU	13	543.965	60.561.000
Germania	NA	16	357.030	82.504.000
Italia	OC	6	301.336	57.888.000
Stati Uniti	SA	1	9.629.091	281.422.000

Loescher Editore - Vietata la vendita e la diffusione

Ricerche, ordinamenti e filtri

Ricerca dati

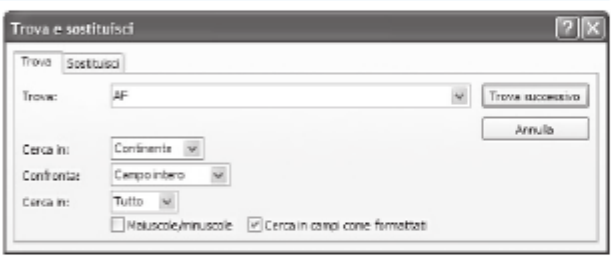
Utilizzando lo strumento ***Trova*** del gruppo *Trova* della barra *Home* si possono compiere semplici operazioni di ricerca sui record; viene presentata la finestra di dialogo *Trova e sostituisci* con attiva la scheda *Trova*, che permette di specificare il valore da ricercare e di scegliere tra diverse opzioni di ricerca; si può effettuare la ricerca su un singolo campo (selezionato nella tabella) o sull'intera tabella e indicare se il valore è tutto o una parte del campo, e se la ricerca va effettuata in tutti i record, o in quelli precedenti o successivi a quello selezionato. Per differenziare tra caratteri maiuscoli e minuscoli bisogna attivare la casella di controllo *Maiuscole/minuscole*.

La casella di controllo *Cerca in campi come formattati* permette di usare per il confronto i valori visualizzati invece di quelli memorizzati nella tabella; il valore visualizzato può differire da quello memorizzato se sono stati specificati formati o maschere di input per la visualizzazione del campo.

La scheda ***Sostituisci*** permette di effettuare sostituzioni, su un record alla volta selezionando prima il pulsante *Trova successivo* e poi il pulsante *Sostituisci*, o su tutti i record selezionando il pulsante *Sostituisci tutto*.

Esempio 8.7

Ricerca nella tabella *Nazioni* delle nazioni africane.



Ordinamento di record

I record di una tabella possono essere ordinati automaticamente in modo crescente o decrescente rispetto ai valori del campo corrente con gli strumenti ***Crescente*** e ***Decrescente*** del gruppo *Ordina e filtra* della barra *Home*. Dopo l'ordinamento, sull'intestazione del campo appare una freccia che indica il verso di ordinamento.

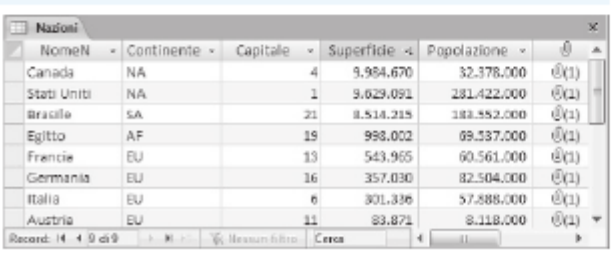
Si può richiedere l'ordinamento in base ai valori di due o più colonne adiacenti selezionando le colonne prima di richiamare lo strumento; i record vengono ordinati in base alla colonna più a sinistra; per valori uguali di questa colonna vengono ordinati in base alla colonna adiacente e così via. Per tutte le colonne viene usato lo stesso verso di ordinamento.

Nota

È possibile annullare l'ordinamento con lo strumento *Cancella tutti gli ordinamenti* del gruppo *Ordina e filtra* della barra *Home*.

Esempio 8.8

Ordinamento della tabella *Nazioni* in ordine decrescente di superficie.



NomeN	Continente	Capitale	Superficie	Popolazione	
Canada	NA	4	9.984.670	32.378.000	(1)
Stati Uniti	NA	1	9.629.091	281.422.000	(1)
Brasile	SA	21	8.514.215	183.552.000	(1)
Egitto	AF	19	998.002	69.537.000	(1)
Francia	EU	13	543.965	60.561.000	(1)
Germania	EU	16	357.030	82.504.000	(1)
Italia	EU	6	301.336	57.888.000	(1)
Austria	EU	11	83.871	8.118.000	(1)

Loescher Editore - Vietata la vendita e la diffusione

Filtri

I **filtri** permettono di ottenere una visualizzazione personalizzata dei dati contenuti nella tabella, selezionando i record che soddisfano determinati criteri e ordinando i record in base al contenuto di più campi, alcuni con ordinamento crescente e altri con ordinamento decrescente.

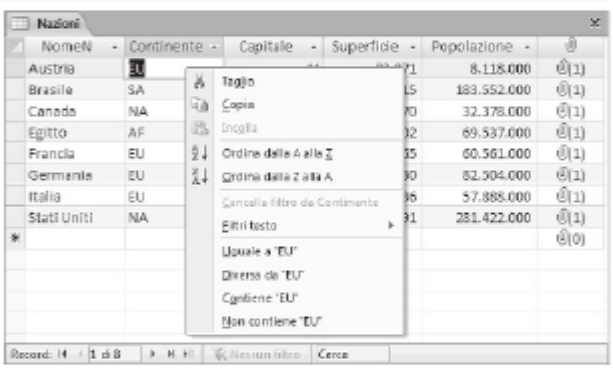
Per applicare un filtro si usano gli strumenti del gruppo ***Ordina e filtra*** della barra *Home*.

Esistono tre tipi di filtri:

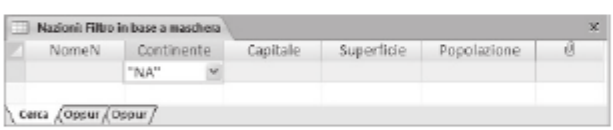
- filtro **per selezione**: dopo aver selezionato il valore di un campo, o una sua parte, si usa lo strumento *Selezione* o si sceglie il criterio dal menu di scelta rapida (si può usare anche lo strumento *Filtro* scegliendo il filtro di testo e impostando il valore);
- filtro **in base a maschera**: permette di ottenere i record che contengono valori specifici in più campi e di specificare valori alternativi; si usa lo strumento *Filtro in base a maschera*; il valore di ciascun campo può essere scelto tramite caselle combinate; i criteri impostati su campi diversi vengono combinati con l'operatore and; per impostare criteri alternativi sullo stesso campo (operatore or) si usano ulteriori maschere che si attivano facendo clic sull'etichetta *Oppure*; terminata la definizione si applica il filtro con lo strumento *Applica filtro*.

Esempio 8.9

Filtro per selezione delle nazioni europee: clic destro su una cella del campo *Continente* con valore *EU*.



Filtro in base a maschera delle nazioni americane (*Continente* con valore NA o CA o SA).



- filtro **avanzato**: consente di specificare criteri complessi e ordinamenti diversi su più campi; lo strumento *Opzioni filtro avanzato*, *Ordinamento/filtro avanzato...* presenta una scheda con una griglia per la definizione dei criteri per il filtro e del tipo di ordinamento per ogni campo; terminata la definizione si applica il filtro con lo strumento *Applica filtro*.

Nota I filtri avanzati si definiscono come le query e possono essere salvati come query.

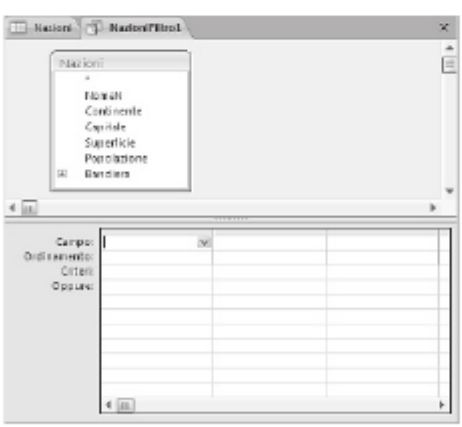


Figura 8.9 *Filtro avanzato*

Un indicatore accanto ai pulsanti di esplorazione segnala che è stato applicato un filtro; facendo clic sulla casella si rimuovono tutti i filtri; per rimuovere i filtri si può usare anche lo strumento *Attiva/disattiva filtro*.

Loescher Editore - Vietata la vendita e la diffusione

Relazioni

Attenzione

Access usa la parola "**relazioni**" per le associazioni tra entità; in questo capitolo la parola relazioni viene usata con questo significato.

Per creare una relazione tra tabelle bisogna aprire la scheda ***Relazioni***, scegliendo lo strumento *Relazioni* del gruppo *Mostra/Nascondi* della barra *Strumenti database*.
Viene presentata la finestra *Mostra tabella* che permette di selezionare le tabelle.

Nota

La finestra *Mostra tabella* in seguito può essere aperta con lo strumento *Mostra tabella* del gruppo *Relazioni* della barra *Struttura*.

Ogni tabella selezionata viene visualizzata nella scheda *Relazioni* con l'elenco dei campi (il campo chiave è evidenziato).

Per definire una relazione tra due tabelle si deve trascinare il campo da correlare da una tabella (tabella primaria) sul campo corrispondente dell'altra tabella (tabella correlata).
Si apre la finestra ***Modifica relazioni*** che permette di impostare il vincolo di integrità referenziale e di scegliere il tipo di join tra le tabelle.

Nota

Il tipo di join è spiegato nel paragrafo 8.3.

Tipi di relazioni

Nella finestra *Modifica relazioni* è indicato il tipo di relazione creata.
Se uno dei campi correlati è una chiave primaria o ha un indice univoco viene creata una relazione **uno a molti**.
Se entrambi i campi correlati sono chiavi primarie o hanno indici univoci viene creata una relazione **uno a uno**.
Se nessuno dei campi è una chiave primaria o ha un indice univoco la relazione è indefinita.

La casella di controllo ***Applica integrità referenziale*** della finestra *Modifica relazioni* permette di stabilire il vincolo di integrità referenziale.

Nota

Perché sia possibile impostare l'integrità referenziale bisogna che il campo correlato della tabella primaria sia una chiave primaria o disponga di un indice univoco e che i campi correlati della tabella correlata abbiano lo stesso tipo di dati.

Se è applicata l'integrità referenziale, sulla linea della relazione vengono utilizzati i simboli 1 e ∞ (molti) per indicare la cardinalità dell'associazione.

La casella ***Aggiorna campi correlati a catena*** fa sì che quando si modifica il valore di una chiave nella tabella primaria venga modificato il valore della chiave esterna in tutti i record correlati.
La casella ***Elimina record correlati*** fa sì che quando si cancella un record nella tabella primaria vengano cancellati anche tutti i record correlati.

Facendo clic sul pulsante *Crea* viene creata la relazione e viene visualizzata una linea che collega i campi corrispondenti delle due tabelle.
Si può eliminare una relazione selezionando la linea e premendo il tasto *Canc*.
Per modificare una relazione basta un doppio clic sulla linea per aprire la finestra *Modifica relazioni*.

Nota

Per eliminare o modificare una relazione si possono usare anche i comandi del menu di scelta rapida relativo alla linea. Per modificare una relazione si può usare lo strumento *Modifica relazioni* del gruppo *Strumenti* della barra *Struttura*.

Loescher Editore - Vietata la vendita e la diffusione

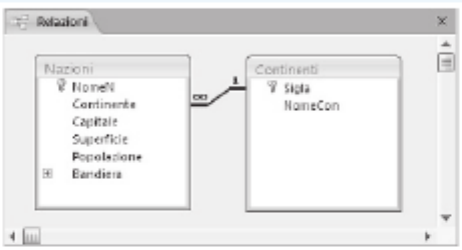
Lo strumento ***Report relazioni*** del gruppo *Strumenti* della barra *Struttura* permette di ottenere un report (un formato stampabile) della scheda *Relazioni*.

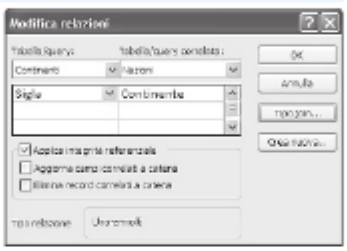
Quando si chiude la scheda *Relazioni* (strumento *Chiudi* del gruppo *Relazioni* della barra *Struttura*) se il layout è stato modificato viene chiesto se si desidera salvarlo.

Se tra due tabelle è stabilita una relazione uno a molti, nel **foglio dati** della tabella dal **lato uno** viene aggiunta a sinistra una casella col simbolo **+**, facendo clic sul simbolo + vengono visualizzate sotto la riga selezionata le righe associate della tabella dal lato molti. Se da una tabella partono più relazioni uno a molti quando si fa clic sul simbolo + viene chiesto quale visualizzare.

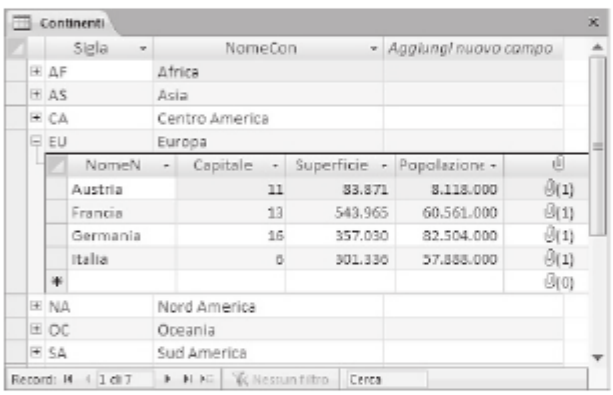
Esempio 8.10

Creazione di una relazione tra le tabelle *Nazioni* e *Continenti*.



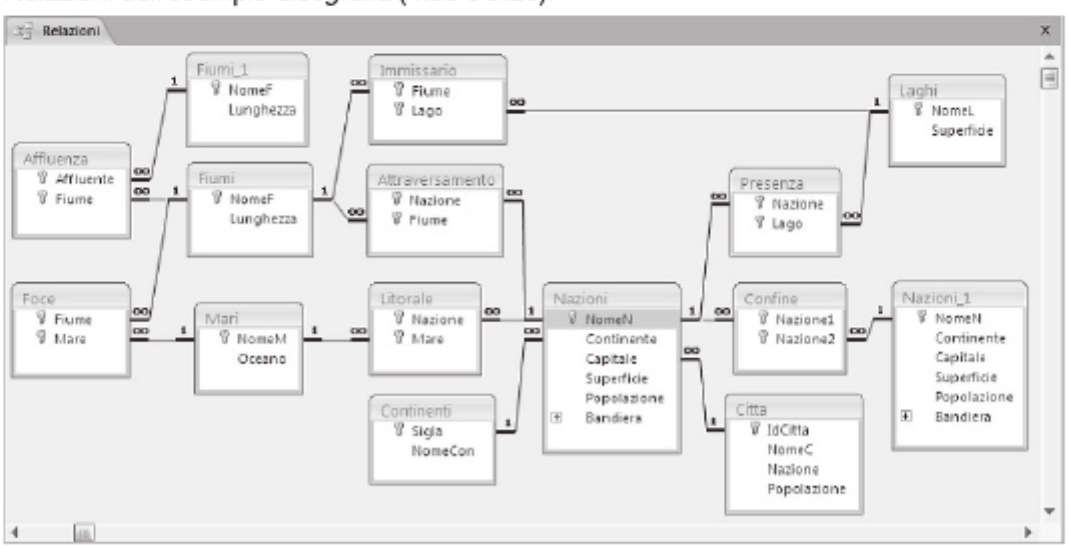


Trascinando il campo *Continente* di *Nazioni* sul campo *Sigla* di *Continenti* (o viceversa) viene creata la relazione (uno a molti).
Nella finestra *Mostra relazioni* la casella *Applica integrità referenziale* è selezionata.
Nel foglio dati di *Continente*, facendo clic sul + accanto al nome di un continente, si vedono le nazioni del continente.



Esempio 8.11

Relazioni dell'esempio *Geografia* (4.38 e 5.20).



Loescher Editore - Vietata la vendita e la diffusione

Query, maschere e report basati su più tabelle

Dopo aver impostato le relazioni tra le tabelle è possibile creare query, maschere e report che estraggono informazioni correlate da diverse tabelle contemporaneamente.

Nota

Se le relazioni sono impostate i collegamenti risultano automatici, altrimenti, se si creano oggetti basati su più tabelle, bisogna creare all'occorrenza i collegamenti necessari tra i dati.

8.3 QUERY

Creazione di query

Le **query di selezione** permettono di eseguire interrogazioni sui dati del database.

Nota

Una query può essere basata su una tabella, su più tabelle o su altre query.

Il dynaset

Il risultato di una query è un set di record, chiamato **dynaset**.
Il dynaset è un insieme dinamico: se variano i dati di origine variano anche i dati restituiti dalla query.
Il dynaset è presentato in un foglio dati su cui è possibile effettuare aggiornamenti che hanno effetto sui dati di origine.

Le query vengono definite in modalità **QBE**.
Per creare una query si usa lo strumento ***Struttura query*** del gruppo *Altro* della barra *Crea*.

Si apre la finestra *Mostra tabella* che permette di selezionare le tabelle su cui basare la query. Facendo clic sul pulsante aggiungi, l'elenco dei campi della tabella viene visualizzato nella scheda di definizione della query su cui diventa attiva la barra ***Struttura***.
Nella griglia di definizione della query è possibile scegliere i campi e i record da includere, ordinare i record ed eseguire calcoli.

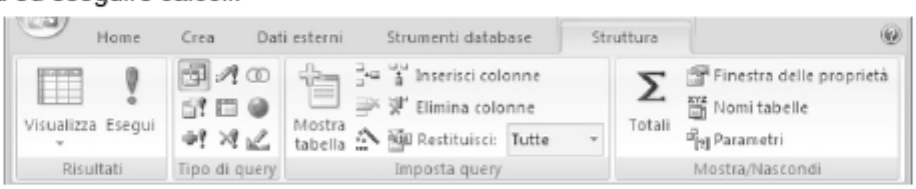


Figura 8.10
La barra degli strumenti Struttura

Creazione guidata query

Si può creare una query con una procedura guidata (strumento ***Creazione guidata Query*** del gruppo *Altro* della barra *Crea*).
Si possono specificare il tipo di query, le tabelle o query da cui selezionare i campi, i calcoli da effettuare, il nome della query e la modalità di apertura (*Struttura* per inserire criteri di selezione o foglio dati con i risultati).

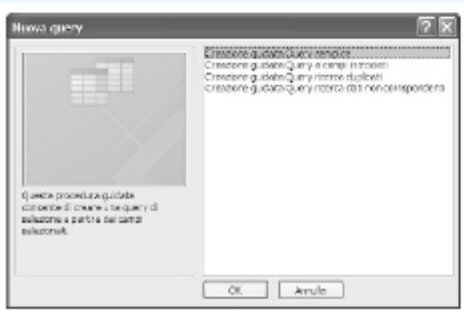


Figura 8.11
Scelta del tipo di query

Loescher Editore - Vietata la vendita e la diffusione

Esecuzione della query

Si può **eseguire** la query con lo strumento ***Esegui*** del gruppo *Risultati* della barra *Struttura*.

Le query già salvate possono essere eseguite scegliendo ***Apri*** dal menu di scelta rapida che si ottiene facendo clic destro sul nome della query nel riquadro di spostamento o semplicemente facendo doppio clic sul nome della query.
L'esecuzione della query presenta il risultato della query in visualizzazione foglio dati.

Impostazione di proprietà delle query

Lo strumento ***Finestra delle proprietà*** del gruppo *Mostra/Nascondi* della barra *Struttura* apre la finestra delle proprietà della query; si possono impostare proprietà per l'intera query e per ogni campo incluso.

Alcune proprietà della query sono:

Valori univoci	permette di omettere i record contenenti dati duplicati;
Primi valori	permette di specificare che vengano restituiti solo un certo numero o una percentuale di record che soddisfano i criteri indicati.

Nota

Il numero o la percentuale di record da restituire possono essere specificati anche con lo strumento *Primi valori* del gruppo *Imposta query* della barra *Struttura*.

Le proprietà di campo possono essere specificate solo per i campi per cui è attiva la casella di controllo *Mostra*:; permettono di scegliere caratteristiche di visualizzazione come un formato o una maschera di input; se non vengono impostate proprietà nella query, vengono utilizzate le proprietà definite nella tabella di origine.

Definizione di query

La scheda di visualizzazione struttura della query contiene la **griglia QBE** per la definizione della query; ogni colonna della griglia permette di specificare le informazioni relative a un campo. La larghezza delle colonne può essere modificata a piacere posizionando il puntatore sul bordo destro della colonna e trascinandolo; facendo doppio clic la larghezza viene adattata automaticamente.

Nota

L'aspetto della griglia può dipendere dall'impostazione di alcuni strumenti del gruppo *Mostra/Nascondi* della barra *Struttura*.

Per **includere i campi** nella griglia è possibile trascinare i campi dall'elenco presente nella parte superiore della scheda (nell'elenco si possono selezionare contemporaneamente più campi tenendo premuto il tasto *Ctrl*) o scegliere un nome dalla casella combinata disponibile nelle celle *Campo:*.

Nella griglia è possibile aggiungere un nuovo campo in fondo o tra gli altri (i campi esistenti si spostano a destra per consentire l'inserimento del nuovo campo), spostare una colonna trascinandola nella posizione desiderata ed eliminare un campo selezionando una colonna e scegliendo lo strumento *Elimina* del gruppo *Record* della barra *Home*.

Se si vogliono includere nella query tutti i campi della tabella è possibile selezionare tutti i campi e trascinarli in gruppo sulla griglia oppure selezionare e trascinare soltanto l'**asterisco** (*).
Se si utilizza l'asterisco anche eventuali campi aggiunti successivamente alla tabella verranno inclusi nella query; non è possibile però utilizzare i campi per definire criteri di selezione o di ordinamento.

Loescher Editore - Vietata la vendita e la diffusione

Volendo utilizzare l'asterisco e anche dei criteri di selezione e ordinamento si possono trascinare tutti i campi con l'asterisco e anche i singoli campi per cui si desidera impostare un criterio, escludendoli però dalla visualizzazione.

Nota

La casella ***Mostra:*** indica se il campo deve essere visualizzato nel risultato della query.

Si può desiderare che un campo non venga visualizzato nel risultato se è stato incluso nella query solo per impostare un criterio di ordinamento o selezione .

Nota

Nella riga ***Ordinamento:*** della griglia si possono impostare i criteri di ordinamento; per ogni campo si può scegliere l'ordinamento crescente o decrescente; l'ordinamento viene eseguito prima sul campo che si trova più a sinistra nella griglia, per valori uguali di questo sul campo successivo e così via (può quindi essere necessario modificare la disposizione dei campi sulla griglia).

La riga ***Criteri:*** permette di definire i criteri di selezione descritti tramite espressioni che usano operatori di confronto e logici.

Criteri impostati per campi diversi sulla stessa riga devono essere soddisfatti contemporaneamente (sono combinati con l'operatore AND); per indicare criteri alternativi (combinati con l'operatore OR) bisogna specificarli sulla riga ***Oppure:***.

Alcune regole per l'impostazione dei criteri:

- l'uguaglianza a un valore specifico può essere controllata scrivendo il valore nella riga (senza virgolette anche per una stringa; le date vanno specificate tra simboli #);
- si possono scrivere espressioni che usano gli operatori relazionali (<, >, <>, <=, >=), l'operatore *Between...And,* l'operatore *In* (separando i valori dell'elenco con punti e virgola) e per stringhe e date l'operatore *Like* (anche con i caratteri jolly tra cui ? che indica un singolo carattere nella posizione corrispondente e * che indica una sequenza di qualsiasi numero di caratteri, anche zero);
- si possono combinare più espressioni con gli operatori logici (*Not*, *And* e *Or*);
- le espressioni possono contenere il richiamo di funzioni.

Alcune funzioni utili sono:

Date()	restituisce la data corrente;
DatePart()	restituisce una parte di una data;
DateAdd()	consente di aggiungere o sottrarre un intervallo di tempo a una data;
Left(), *Right()*, *Mid()*	consentono di estrarre caratteri a sinistra, a destra o al centro di una stringa.

Per scrivere le espressioni si può usare il ***Generatore di espressioni***.
Si può avviare il *Generatore di espressioni* con lo strumento *Generatore* del gruppo *Imposta query* della barra *Struttura* o scegliendo *Genera...* dal menu di scelta rapida che si apre facendo clic con il pulsante destro del mouse nella casella in cui si deve scrivere l'espressione.

Si possono vedere le funzioni disponibili scegliendo nel primo riquadro *Funzioni*, *Funzioni predefinite*, nel secondo la categoria e nel terzo la funzione desiderata.

Nota

Attributi derivati

Gli **attributi derivati** si definiscono scrivendo l'espressione corrispondente in una casella *Campo:* vuota.

Loescher Editore - Vietata la vendita e la diffusione

L'espressione può includere nomi di campo racchiusi tra parentesi quadre ([ ]).
Per concatenare più stringhe di testo si usa l'operatore &.
Possono essere utili le funzioni:

IIF()	restituisce due valori diversi in base a una condizione,
Sum()	calcola la somma dei valori di un campo o di un'espressione indicata come argomento, per un gruppo di record;
DLookup()	permette di cercare il valore di un campo in una tabella specificata.

Quando si preme *Invio* dopo aver scritto l'espressione, nella cella *Campo:* viene aggiunto un nome predefinito concatenato all'espressione; se il campo deve essere visualizzato è opportuno sostituire il nome con uno più descrittivo, poiché viene utilizzato come intestazione della colonna nel foglio dati.

Query parametriche

Per creare **query parametriche** basta scrivere il messaggio di richiesta del parametro tra parentesi quadre nella cella *Criteri:* del campo.

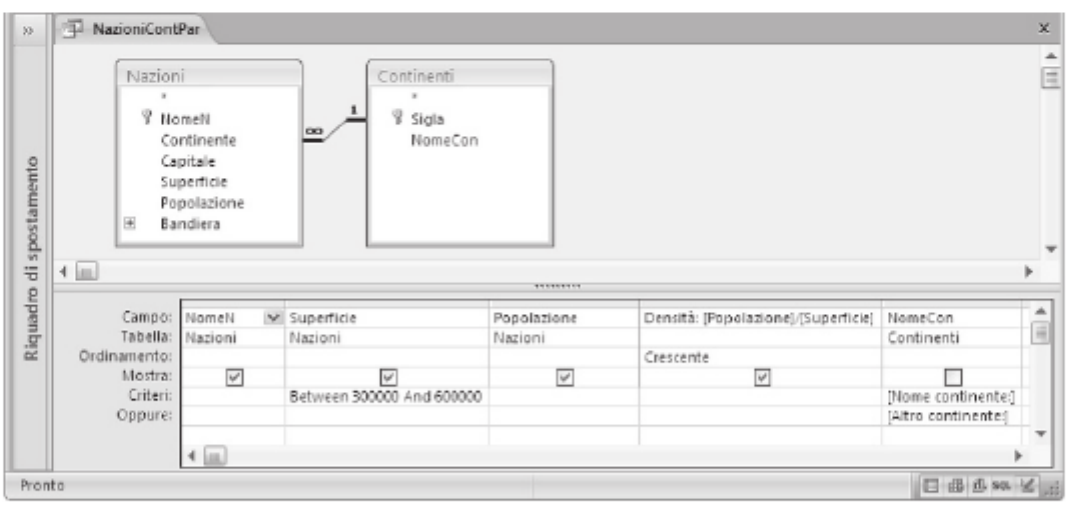
Nota

Se i parametri vengono usati in espressioni complesse può essere utile definire i parametri con lo strumento *Parametri* del gruppo *Mostra/Nascondi* della barra *Struttura*; si apre la finestra *Parametri query* in cui specificare il nome e tipo del parametro.

Durante l'esecuzione la query presenta la finestra di dialogo *Immettere valore parametro* per ogni parametro necessario.

Esempio 8.12

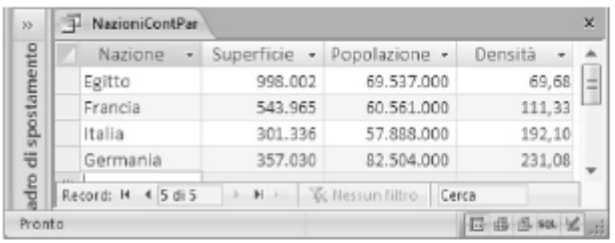
Nome, superficie, popolazione e densità delle nazioni con superficie compresa tra 300.000 e 600.000 di due continenti scelti dall'utente al momento dell'esecuzione, in ordine crescente di densità.



Nella griglia sono inseriti i campi *NomeN*, *Superficie* e *Popolazione* di *Nazioni* e *NomeCon* di *Continenti*; *NomeCon* non è visualizzato nel foglio dati (casella *Mostra:* disattivata).
Densità è un attributo derivato calcolato come *Popolazione / Superficie*; su questa colonna è impostato l'ordinamento crescente.
Nella colonna *Superficie* è impostato il criterio per l'intervallo di valori con l'operatore *Between*.
Nella colonna *NomeCon* è impostato il criterio di uguaglianza a uno di due nomi (operatore *or*) inseriti dall'utente al momento dell'esecuzione.

Loescher Editore - Vietata la vendita e la diffusione

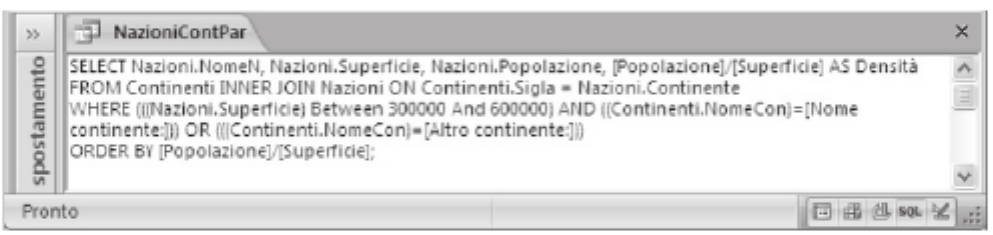
Eseguendo la query viene presentata una finestra di dialogo per ogni parametro; il foglio dati presenta il risultato della query.



SQL

È possibile vedere il **codice SQL** della query con lo strumento *Visualizza*, ***Visualizzazione SQL*** del gruppo *Risultati* della barra *Struttura*.

Codice SQL della query dell'esempio 8.12.



Lo strumento *Visualizzazione SQL* permette anche di scrivere una query direttamente in SQL.

Nota L'unione di tabelle o query di selezione è definibile solo scrivendo il codice SQL.

Query con calcoli e raggruppamenti

Le **funzioni di aggregazione** e i **raggruppamenti** possono essere definiti nella riga ***Formula:*** che viene aggiunta alla griglia con lo strumento *Totali* del gruppo *Mostra/Nascondi* della barra *Struttura*.

Alcune valori utilizzabili nella riga ***Formula:*** sono:

Raggruppamento	è il valore di default; permette di raggruppare i record in base al valore del campo;
Dove	specifica la condizione che devono soddisfare i record su cui eseguire i calcoli;
Conteggio	calcola il numero di valori nel campo, senza contare i valori nulli;
Somma	calcola il totale dei valori nel campo;
Media	calcola la media dei valori nel campo;
Min	calcola il valore minimo presente nel campo;
Max	calcola il valore massimo presente nel campo;

Per **eseguire calcoli su tutti i record** della tabella nessuna casella deve essere impostata al valore *Raggruppamento*; ogni casella deve invece contenere il tipo di formula desiderato.
Per **eseguire calcoli su gruppi di record** uno o più campi devono essere impostati al valore *Raggruppamento* (i campi vengono raggruppati da sinistra a destra, nell'ordine in cui sono visualizzati) e altri a formule di calcolo.

Loescher Editore - Vietata la vendita e la diffusione

Per quanto riguarda i **criteri di selezione**:

- per raggruppare solo record che soddisfano i criteri bisogna impostare i criteri sul campo col valore *Raggruppamento*;
- per calcolare i risultati della formula su tutti i record e poi visualizzare soltanto i risultati che soddisfano i criteri bisogna impostare i criteri sul campo con la formula di calcolo (clausola HAVING di SQL);
- per limitare i record su cui eseguire i calcoli della formula bisogna selezionare il valore *Dove* nella cella *Formula:* del campo per cui si desidera impostare il criterio di selezione e disattivare la casella di controllo *Mostra:* relativa.

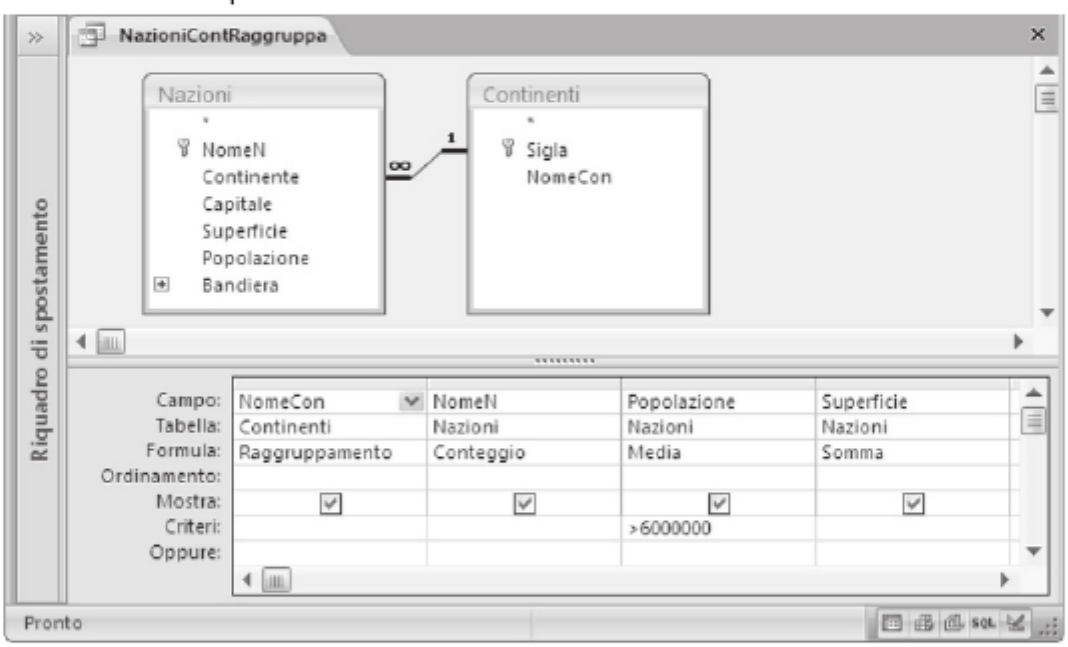
Nel foglio dati della query le intestazioni delle colonne riportano il tipo di calcolo eseguito; si possono ridenominare le colonne modificando la proprietà *Etichetta* del campo.

Nota

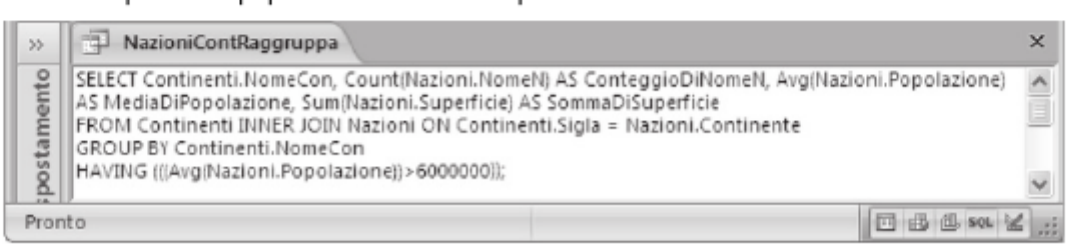
I dati restituiti da una query con formule non possono essere modificati.

Esempio 8.13

Numero di nazioni, superficie totale e media della popolazione delle nazioni per i continenti in cui la media è superiore a 6.000.000.



Formula: nel campo *Continenti* è impostata a *Raggruppamento* per raggruppare le nazioni per continente; nei campi *NomeN*, *Popolazione* e *Superficie* è impostata rispettivamente a *Conteggio* (numero delle nazioni), *Media* (calcolo della popolazione media) e *Somma* (calcolo della superficie totale); il criterio nel campo *Popolazione* visualizza solo i dati dei continenti per cui la popolazione media è superiore a 6.000.000.



Loescher Editore - Vietata la vendita e la diffusione

Query a campi incrociati

Le **query a campi incrociati** permettono di visualizzare dati di riepilogo in una forma compatta; i dati vengono raggruppati in base ai campi elencati nella prima riga e nella prima colonna del foglio dati prodotto; nelle caselle all'incrocio tra una riga e una colonna viene visualizzato il valore di riepilogo (per esempio somma, conteggio o media) di un campo della tabella.

Nota

Una query a campi incrociati restituisce un'immagine istantanea dei dati; non è possibile apportare modifiche ai dati visualizzati.

Dopo aver scelto le tabelle in visualizzazione struttura della query, si seleziona lo strumento ***Tipo query: a campi incrociati*** del gruppo *Tipo query* della barra *Struttura*.
Nella griglia vengono visualizzate le righe *Formula:* e *Campi incrociati:*.

La casella ***Formula:*** di ogni campo aggiunto nella griglia viene automaticamente impostata al valore *Raggruppamento*.

La casella ***Campi incrociati:*** del campo che si desidera usare come intestazione di colonna deve essere impostata a ***Intestaz. colonna*** (lasciando la cella *Formula:* impostata al valore *Raggruppamento*; i valori del campo costituiranno le intestazioni delle colonne del foglio dati.

Nota

È possibile modificare l'aspetto delle colonne del foglio dati di una query a campi incrociati impostando la proprietà *Intestazioni colonne* nella finestra delle proprietà della query.

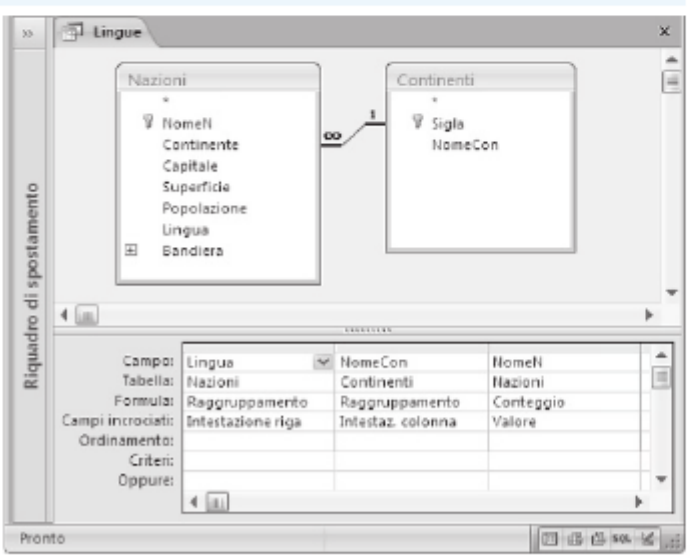
La casella ***Campi incrociati:*** dei campi che si desidera usare come intestazione di riga deve essere impostata a ***Intestazione riga*** (lasciando la cella *Formula:* impostata al valore *Raggruppamento*); ogni campo costituisce una colonna, i cui valori sono le intestazioni di riga del foglio dati.

Nel campo da cui si desidera estrarre i valori di riepilogo bisogna impostare la formula di calcolo desiderata nella cella ***Formula:*** e impostare la cella ***Campi incrociati:*** a ***Valore***.

Nella query si possono includere altri campi usati solo per raggruppare, ordinare o impostare criteri, che non devono essere visualizzati, e devono avere la casella *Campi incrociati:* impostata a *(Non visualizzato)*.

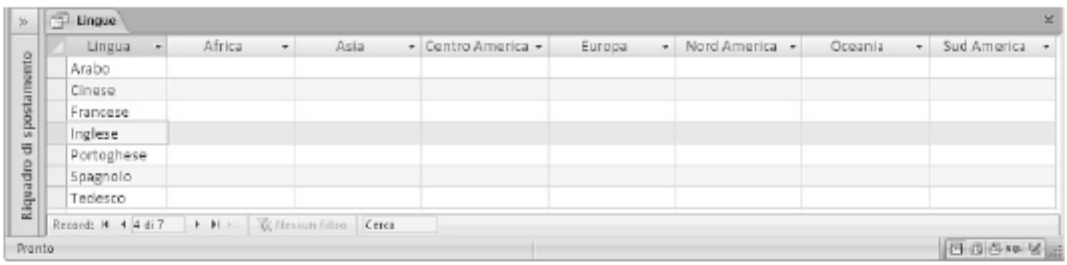
Esempio 8.14

Aggiungendo alla tabella *Nazioni* il campo *Lingua* è possibile calcolare il numero di nazioni con la stessa lingua per ogni continente.



Loescher Editore - Vietata la vendita e la diffusione

L'output della query ha il seguente schema:



Query basate su più tabelle e linee di join

Si possono realizzare query basate su più tabelle o anche su altre query.
Dopo aver scelto le tabelle nella finestra *Mostra tabella* si possono trascinare nella griglia campi provenienti da tabelle diverse.
I nomi delle tabelle sono visibili nella griglia alla riga ***Tabella***: se è attivo lo strumento *Nomi tabelle* del gruppo *Mostra/Nascondi* della barra *Struttura*.

Una query basata su più tabelle funziona correttamente se le tabelle sono collegate da **linee di join** che stabiliscono le correlazioni tra i dati; la linea di join collega i campi correlati tra gli elenchi dei campi delle tabelle.

Nota Se non vengono create le linee di join si ottiene il prodotto cartesiano delle tabelle.

Se tra le tabelle esiste una relazione la linea di join viene creata automaticamente; se è applicata l'integrità referenziale vengono visualizzati anche i simboli 1 e ∞ che indicano il tipo di relazione.
Viene creata automaticamente una linea di join anche se le tabelle contengono un campo con lo stesso nome e tipo di dati e uno dei due è una chiave primaria.
Si può creare una linea di join trascinando il campo di una tabella sul campo da correlare dell'altra tabella.

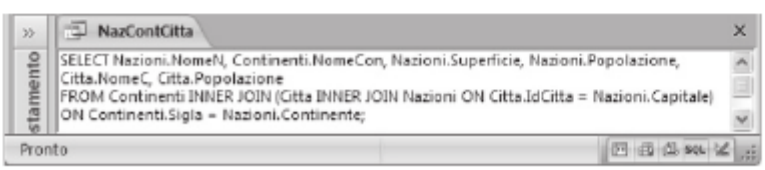
Per default i join sono interni (equi-join); si può modificare il **tipo di join** (scegliendo join sinistro o join destro) dalla finestra ***Proprietà join*** che si apre facendo doppio clic sulla linea di join (o dal menu di scelta rapida relativo alla linea di join).

Nota Per le relazioni si può scegliere il tipo di join nella finestra *Proprietà join* a cui si accede con il pulsante *Tipo join...* della finestra *Modifica relazioni*.

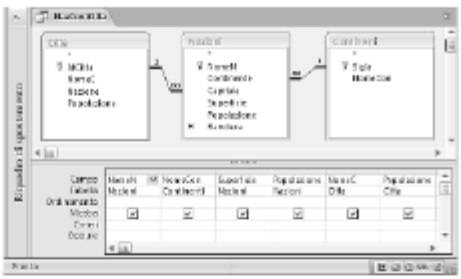
Figura 8.12 *Proprietà join della linea di join della relazione uno a molti tra Continenti e Nazioni*

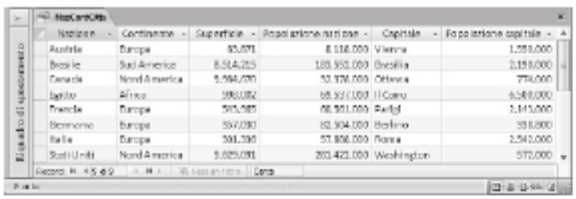
Esempio 8.15

Elenco delle nazioni con nome, continente, superficie, popolazione, capitale e popolazione della capitale.



Loescher Editore - Vietata la vendita e la diffusione





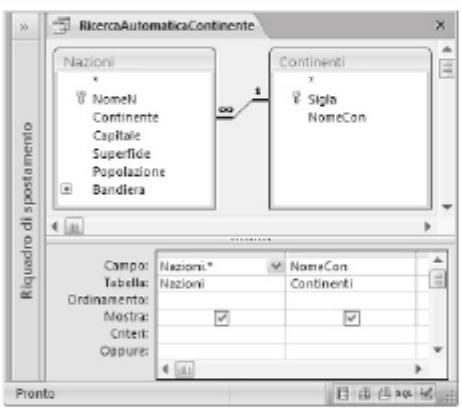
Ricerca automatica di informazioni

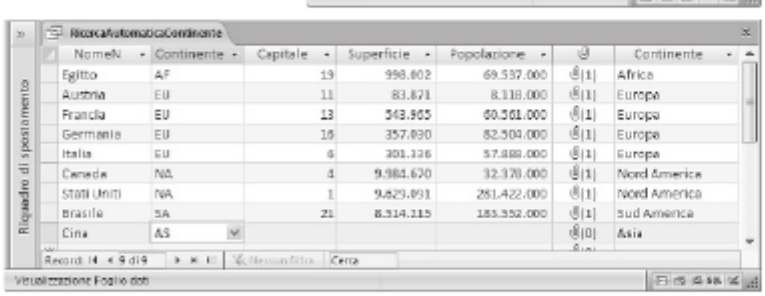
La **ricerca automatica** è disponibile in una query basata su due tabelle tra cui esiste una relazione uno a molti se il campo di join sul lato uno della relazione ha un indice univoco. Nella query va incluso il campo di join proveniente dalla tabella dal lato molti.
Quando in un record del dynaset si aggiunge o modifica il valore del campo di join del lato molti vengono visualizzati automaticamente i campi della tabella del lato uno inclusi nella query (se esiste un record corrispondente).

Esempio 8.16

Query *RicercaAutomaticaContinente*, basata sulle tabelle *Nazioni* e *Continenti*, per creare un foglio dati con tutti i dati della tabella *Nazioni* in cui la sigla del continente viene decodificato automaticamente.

Nella griglia si può portare l'asterisco dall'elenco dei campi della tabella *Nazioni* per includere tutti i campi della tabella, e il campo *NomeCon* dalla tabella *Continenti*.
Quando sul foglio dati si modifica, o si inserisce in un nuovo record, la sigla del continente, il nome del continente compare automaticamente.





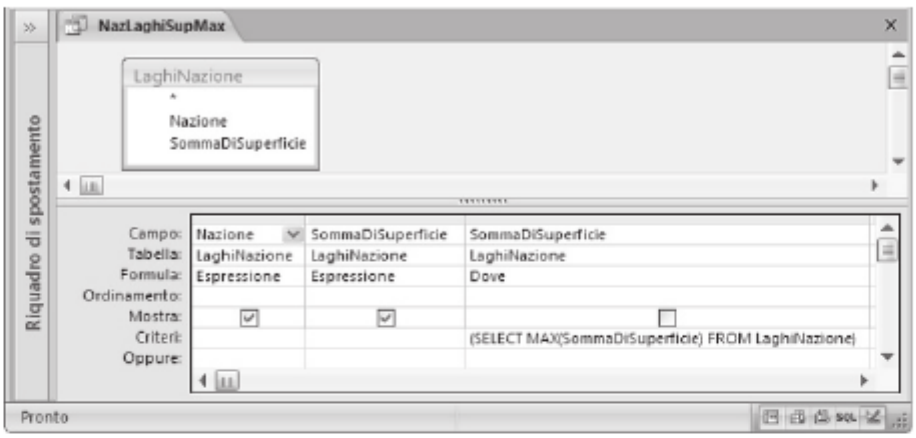
Query come viste

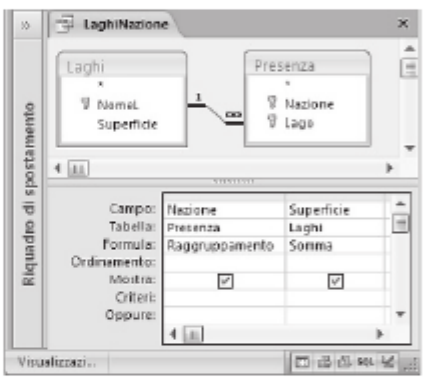
In Access le **viste** vengono realizzate mediante **query**.
È possibile basare una query su altre query; una query può costituire anche la base per una maschera o un report; in questo modo è possibile limitare l'accesso ai dati, rendendo visibili solo determinati campi o record.

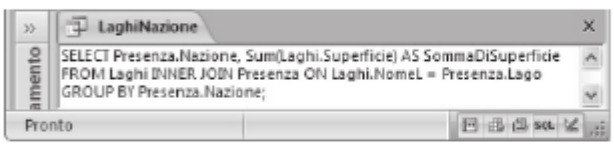
Loescher Editore - Vietata la vendita e la diffusione

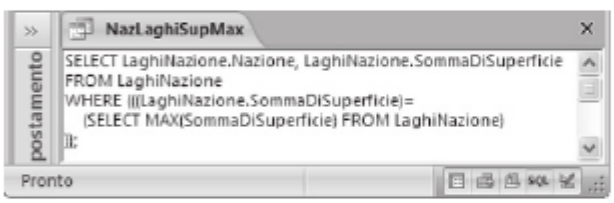
Esempio 8.17

Trovare la nazione con la superficie totale dei laghi maggiore. Per semplificare la query è possibile utilizzare una query che elenca per ogni nazione il numero di laghi e la superficie totale dei laghi.









Query di comando

Le **query di comando** permettono di eseguire operazioni di manipolazione dei dati (DML).

Per creare una query, dopo aver scelto le tabelle in visualizzazione struttura della query, si seleziona lo strumento relativo a una query di comando dal gruppo ***Tipo query*** della barra *Struttura*.

Le query di comando vengono eseguite quando vengono aperte.
L'esecuzione delle query di comando deve essere consentita: bisogna visualizzare la barra dei messaggi con lo strumento *Barra dei messaggi* del gruppo *Mostra/Nascondi* della barra *Strumenti database*, fare clic sul pulsante *Opzioni...* e nella finestra *Opzioni di protezione Microsoft Office* selezionare la voce ***Attiva il contenuto***.

Quando si manda in esecuzione una query di comando, viene chiesta conferma con una finestra di dialogo che mostra quanti record saranno modificati.

L'esecuzione non mostra il foglio dati risultante; per vederlo si usa lo strumento *Visualizza*, *Visualizzazione Foglio dati* del gruppo *Risultati* della barra *Struttura*.

Nota

A volte per vedere gli effetti della query bisogna aprire le tabelle modificate in modalità foglio dati.

Si possono nascondere le query di comando (per evitare che vengano eseguite per errore) scegliendo *Nascondi in questo gruppo* dal menu di scelta rapida relativo alla query nel

Loescher Editore - Vietata la vendita e la diffusione

riquadro di spostamento. Per mostrare di nuovo le query nascoste si può scegliere *Opzioni di spostamento...* dal menu di scelta rapida del riquadro di spostamento e selezionare la casella di controllo *Mostra oggetti nascosti* e poi scegliere *Scopri in questo gruppo* dal menu di scelta rapida relativo alla query.

Query di creazione tabella

Una **query di creazione tabella** crea una nuova tabella con i record selezionati dalla query. Viene chiesto il nome della nuova tabella e nel corso dell'esecuzione indicato quanti record verranno inclusi.

Importazione dati

Si possono **importare dati** con gli strumenti del gruppo *Importa* della barra *Dati esterni*, si possono importare dati per esempio da un file di testo, da un file XML, da una tabella di Excel, da un documento HTML e da altri formati di database.

Query di accodamento

Una **query di accodamento** aggiunge a una tabella i record selezionati dalla query (che ovviamente devono essere compatibili).
La finestra di dialogo *Accodamento* chiede la tabella destinazione.
Nella griglia di definizione è presente la riga *Accoda a:*
Bisogna indicare nella casella *Campo:* il campo della tabella di origine da cui prelevare i dati e nella casella *Accoda a:* il nome del campo della tabella di destinazione; se nella tabella di destinazione esiste un campo con lo stesso nome di quello della tabella di origine l'impostazione è automatica.
Se nella casella *Campo:* si usa l'asterisco vengono accodati automaticamente tutti i campi con i nomi corrispondenti.
I campi della tabella di destinazione per cui non è previsto l'accodamento vengono lasciati vuoti.

Query di aggiornamento

Una **query di aggiornamento** modifica uno o più campi di una tabella nei record selezionati dalla query.
Nella griglia di definizione è presente la riga *Aggiorna a:* in cui si definisce il valore o l'espressione da utilizzare per modificare il campo corrispondente.

Query di eliminazione

Una **query di eliminazione** elimina da una tabella i record selezionati dalla query.
Nella griglia di definizione è presente la riga *Elimina:*.
Si può trascinare in una casella *Campo:* l'asterisco dalla tabella su cui effettuare l'operazione; nella casella *Elimina:* corrispondente compare il valore *Da*.
Per i campi di altre tabelle inseriti nella griglia per definire criteri di selezione la casella *Elimina:* deve essere impostata a *Dove*.

Query di definizione dati

Le **query di definizione dati** permettono di creare, modificare o eliminare oggetti del database (comandi DDL).
Possono essere realizzate solo scrivendo esplicitamente il codice SQL.

Loescher Editore - Vietata la vendita e la diffusione

8.4 MASCHERE

Le **maschere** permettono di creare finestre con layout personalizzati per organizzare i dati nel modo desiderato.

Creazione di maschere

Per creare maschere si usano gli strumenti del gruppo *Maschere* della barra *Crea*.

Lo strumento ***Maschera*** crea automaticamente una maschera basata sulla tabella selezionata, con un'etichetta e una casella di testo per ogni campo della tabella disposti in colonna uno sotto l'altro (disposizione a colonne); possono essere creati controlli diversi da caselle di testo in base al tipo di dati dei campi e alle proprietà impostate.

Nota

Se sulla tabella sono definite delle relazioni vengono create automaticamente maschere complesse (vedi a pagina 229).

La maschera è in ***Visualizzazione Layout***; sono visibili i dati ed è possibile modificare la posizione e l'aspetto dei controlli mediante le barre *Disponi e Formato* (in primo piano).
Lo strumento *Visualizza* del gruppo *Visualizzazioni* della barra *Formato* permette di passare alle modalità ***Visualizzazione Struttura*** (con la barra *Struttura*) in cui è possibile modificare la struttura della maschera cambiando numero e tipo di controlli, alla modalità ***Visualizzazione Maschera*** in cui si usa la maschera per accedere ai dati (scorrere i record, modificarli e aggiungerne di nuovi) e alla ***Visualizzazione Foglio dati*** in cui si ottiene il foglio dati della tabella su cui è basata la maschera.

Altri strumenti del gruppo *Maschere* sono:

Maschera divisa mostra sulla stessa scheda la maschera e il foglio dati; in visualizzazione maschera si può operare sui dati sia dalla maschera che dal foglio dati;

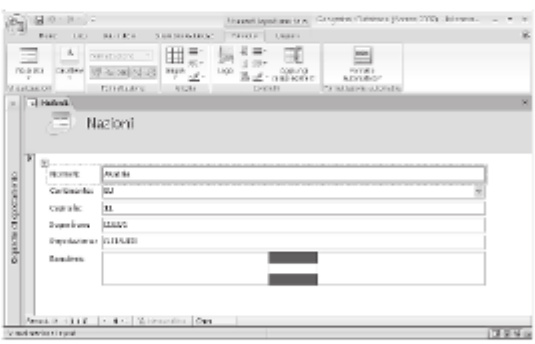


Figura 8.13 *Maschera creata con: Maschera*

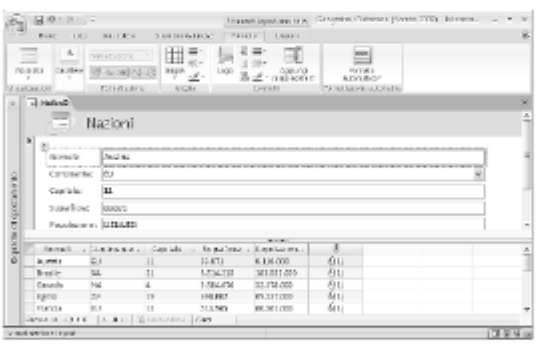


Figura 8.14 *Maschera creata con: Maschera divisa*

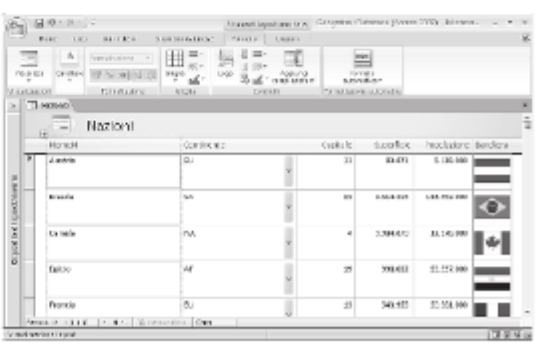


Figura 8.15 *Maschera creata con: Più elementi*

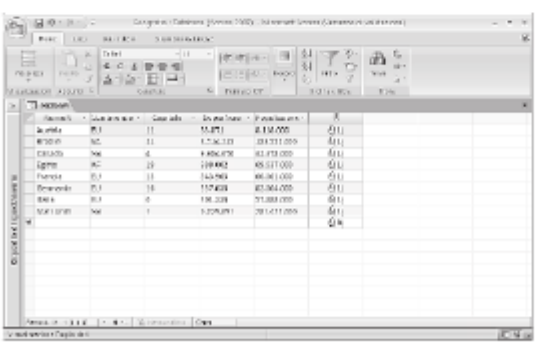


Figura 8.16 *Maschera creata con: Foglio dati*

Loescher Editore - Vietata la vendita e la diffusione

Più elementi crea una disposizione tabellare con i record uno per riga in visualizzazione layout;
Maschera vuota apre una scheda vuota e la finestra *Elenco campi* da cui è possibile scegliere i campi da inserire nella scheda, trascinandoli;
Altre maschere, ***Foglio dati*** mostra il foglio dati della tabella su cui è basata la maschera;
Struttura maschera permette di creare o modificare una maschera in visualizzazione struttura.

Creazione guidata di una maschera

Si può creare una maschera con una procedura guidata (strumento *Altre maschere*, ***Creazione guidata Maschera*** del gruppo *Maschere* della barra *Crea*).

Si devono specificare:

- i **campi** da inserire selezionandoli da tabelle e query; i campi di una tabella sono visualizzati nel riquadro di sinistra e possono essere aggiunti al riquadro a destra col pulsante >; il pulsante >> li aggiunge tutti;
- la **disposizione** dei controlli della maschera tra quattro modelli possibili:

A colonne un record per volta con i campi disposti in colonna uno sotto l'altro;
Tabulare un record per volta ciascuno su una riga;
Foglio dati ha l'aspetto di un foglio dati;
Giustificato i campi sono disposti prima uno accanto all'altro finché c'è spazio e poi su più righe.

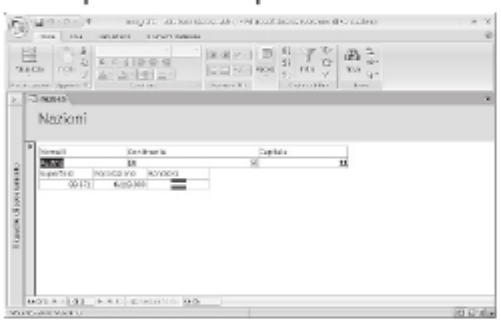


Figura 8.17 *Maschera con modello Giustificato*

- lo **stile** dei controlli, del testo e dello sfondo della maschera tra 26 stili predefiniti;
- il **nome** della maschera e la **modalità di visualizzazione** (*Visualizzazione Maschera* o *Visualizzazione Struttura*).

Proprietà della maschera

In visualizzazione struttura lo strumento ***Finestra delle proprietà*** del gruppo *Strumenti* della barra *Struttura* apre la finestra delle proprietà della maschera.

Le proprietà sono divise in schede.

- la scheda ***Formato*** contiene le proprietà relative alla presentazione della maschera; le proprietà principali sono:

Visualizzazione predefinita
permette di stabilire l'aspetto della maschera al momento dell'apertura:
Maschera singola disposizione a colonne (un record per volta con i campi disposti in colonna uno sotto l'altro);
Maschere continue disposizione tabellare (un record per volta ciascuno su una riga);
Foglio dati la maschera viene aperta in visualizzazione foglio dati;
Dividi maschera la maschera viene aperta insieme al foglio dati;

Loescher Editore - Vietata la vendita e la diffusione

Consenti Visualizzazione maschera, *Consenti Visualizzazione Foglio dati*, *Consenti Visualizzazione Tabella Pivot*, *Consenti Visualizzazione Grafico Pivot*, *Consenti Visualizzazione Layout*: ciascuno permette di specificare se la maschera può essere visualizzata in tale modalità;
Pulsanti spostamento indica se deve essere presente la barra con i pulsanti di spostamento;
Didascalia spostamento testo da mostrare nella barra con i pulsanti di spostamento;
Immagine permette di impostare una immagine per lo sfondo;
Tipo immagine indica se l'immagine di sfondo deve essere incorporata o collegata;
Espansione immagine indica se l'immagine di sfondo deve apparire una volta sola (valore *No*) o essere ripetuta più volte (*Si*) per riempire tutta la maschera;

- la scheda ***Dati*** contiene le proprietà relative all'origine dei dati; alcune proprietà sono:

Origine record indica la tabella o la query su cui si basa la maschera;
Filtro permette di impostare un filtro sui dati;
Filtra al caricamento indica se il filtro specificato nella proprietà *Filtro* deve essere attivato al caricamento;

- la scheda ***Evento*** permette di specificare le macro che devono essere eseguite al verificarsi degli eventi;

- la scheda ***Altro*** contiene proprietà aggiuntive; alcune proprietà sono:

Popup il valore *Sì* indica che la finestra della maschera rimane sempre in primo piano;
A scelta obbligatoria il valore *Sì* rende la maschera una finestra di dialogo a scelta obbligatoria (non permette l'interazione dell'utente con altre finestre dell'applicazione finché non viene chiusa); la finestra non si può spostare nè ridimensionare.

Nota

La scheda *Tutte* visualizza tutte le proprietà insieme.

Stampa di una maschera

Si può **stampare** un maschera con il comando *Stampa* dal pulsante *Office*.
Se la maschera si trova in visualizzazione struttura o in visualizzazione maschera si ottiene la disposizione tabellare di più record (come per il valore *Maschere continue* della proprietà *Visualizzazione predefinita*); se la maschera si trova in visualizzazione foglio dati si ottiene la stampa del foglio dati.

Operazioni sui record tramite le maschere

In visualizzazione maschera si può usare la maschera per gestire i **dati** della tabella su cui si basa la maschera.

Origine record

Si può usare una maschera basata su una tabella per eseguire operazioni su un'altra tabella con lo stesso tipo di campi, cambiando la tabella di origine nella proprietà ***Origine record*** della maschera.

Loescher Editore - Vietata la vendita e la diffusione

Le maschere permettono di eseguire sui record le stesse operazioni possibili sul foglio dati. Se la tabella contiene dati, i controlli associati a campi della tabella visualizzano i dati corrispondenti. I **pulsanti di spostamento** sull'ultima riga della maschera permettono di posizionarsi sul record desiderato.

Si possono inserire record, modificarli o cancellarli ed effettuare ricerche e sostituzioni, applicare ordinamenti e filtri con gli strumenti dei gruppi *Trova* e *Ordina e filtra* della barra *Home*.

IMPEDIRE OPERAZIONI

Si può impedire l'applicazione di filtri, la modifica di dati, l'eliminazione o l'inserimento di record impostando le proprietà booleane della maschera ***Consenti filtri***, ***Consenti modifiche***, ***Consenti eliminazioni*** e ***Consenti aggiunte***.

È possibile spostarsi tra i campi della maschera con i tasti di tabulazione; l'ordine di tabulazione dipende dall'ordine di creazione dei controlli, ma è possibile modificarlo in visualizzazione struttura con lo strumento ***Ordine di tabulazione*** del gruppo *Layout controllo* della barra *Disponi*.

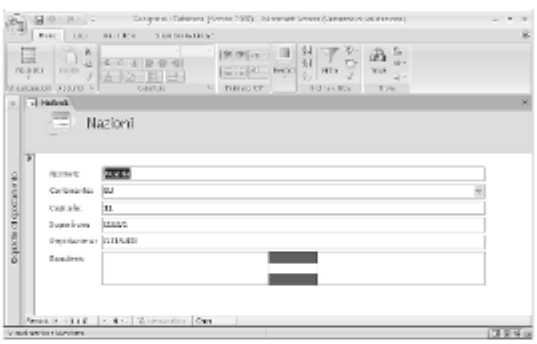


Figura 8.18 *Visualizzazione maschera con disposizione a colonne*

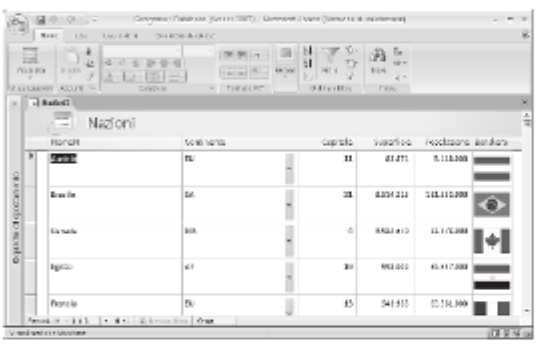


Figura 8.19 *Visualizzazione maschera con disposizione tabellare*

I CONTROLLI

Gli oggetti che possono essere posti nella maschera sono detti **controlli**.
I controlli possono essere creati e modificati in visualizzazione struttura della maschera.

I controlli si dividono in **associati**, che si collegano ad un campo della tabella di origine, **non associati** che servono solo per completare la struttura della maschera e **calcolati**, che hanno come fonte di dati un'espressione.

> Le caselle di testo possono essere controlli associati, non associati o calcolati; le etichette sono sempre controlli non associati; spesso le etichette sono in relazione con altri controlli, in questo caso si usa l'espressione etichetta allegata (associata al controllo).
>
> **Nota**

Si possono creare i controlli con gli strumenti del gruppo ***Controlli*** della barra *Struttura*, attiva quando si è in visualizzazione struttura.

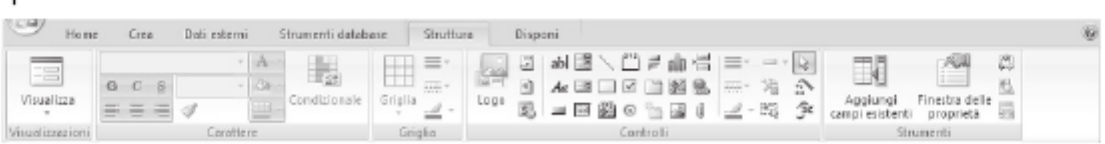


Figura 8.20 *La barra degli strumenti Struttura*

Per **creare** un controllo si può selezionare lo strumento corrispondente, posizionare il cursore sulla maschera nel punto in cui si vuole disegnare il controllo e trascinare fino a raggiungere le dimensioni volute.

Loescher Editore - Vietata la vendita e la diffusione

Se è attivo lo strumento *Usa creazioni guidate Controllo* al momento della creazione del controllo può essere avviata una procedura guidata per la scelta delle caratteristiche. **Nota**

Creazione di caselle di testo associate

Una **casella di testo associata** è collegata a un campo della tabella e permette di gestire i dati del campo. Si può creare una casella di testo associata trascinando il campo dall'elenco dei campi della tabella (la finestra *Elenco campi* che si ottiene con lo strumento *Aggiungi campi esistenti* del gruppo *Strumenti* della barra *Struttura*).
Ogni casella di testo dispone di una **etichetta allegata**, costituita dal nome del campo, o dal nome indicato nella proprietà *Etichetta* del campo della tabella, se definito.

Spostamento e modifica di controlli

Per spostare o modificare un controllo bisogna selezionarlo.
Per **selezionare** un controllo basta fare clic su di esso; si possono selezionare più controlli contemporaneamente trascinando il puntatore fino a comprendere tutti i controlli desiderati, tenendo premuto il tasto *Shift* mentre si fa clic o si trascina il mouse se i controlli non sono adiacenti. Si possono selezionare tutti i controlli con lo strumento *Seleziona tutto* del gruppo *Controlli* della barra *Struttura*.

Se un controllo dispone di un'etichetta allegata, è possibile agire sia sul controllo che sull'etichetta; per **spostare** insieme etichetta e controllo basta selezionare uno dei due e trascinare il bordo (il puntatore del mouse assume la forma di due frecce a due punte perpendicolari); per spostare solo l'etichetta o solo il controllo si trascina il relativo quadratino nell'angolo in alto a sinistra.

Per **modificare le dimensioni** di un controllo si trascina una delle maniglie di ridimensionamento: i quadratini nei vertici e a metà dei lati (il puntatore assume la forma di una freccia a due punte).

Per **eliminare** un controllo selezionato si può usare il tasto *Canc* o scegliere *Elimina* dal suo menu di scelta rapida. Se solo l'etichetta è selezionata viene cancellata l'etichetta ma non il controllo.

Spostamenti e **copie** di controlli si possono effettuare con gli strumenti *Taglia*, *Copia* e *Incolla* del gruppo *Appunti* della barra *Home*.

Gli strumenti del gruppo ***Carattere*** della barra *Formato* permettono di modificare il font e lo stile del carattere, il colore del testo e dello sfondo, l'allineamento; lo strumento *Condizionale* permette di applicare una formattazione in base al verificarsi di una condizione (per esempio di applicare un colore diverso ai valori maggiori di un valore dato).

Lo strumento ***Formato automatico*** del gruppo *Formattazione automatica* permette la scelta di uno stile predefinito per la maschera o l'avvio di una procedura guidata per l'impostazione dello stile.

Figura 8.21 *La barra degli strumenti Formato*

Gli strumenti della barra ***Disponi*** permettono di controllare allineamenti, dimensioni e spaziature di un gruppo di controlli per ottenere un aspetto uniforme. Gli strumenti *Tabulare* e *In pila* del gruppo *Layout controllo*: allineano i controlli, rispettivamente, in una tabella di due righe con le etichette come intestazione e i campi nella seconda riga, o come insieme di coppie etichetta/controllo una sotto l'altra.

Per disporre più facilmente i controlli si possono usare griglie e righelli (strumenti *Griglia* e *Righello* del gruppo *Mostra/Nascondi*). **Nota**

Loescher Editore - Vietata la vendita e la diffusione

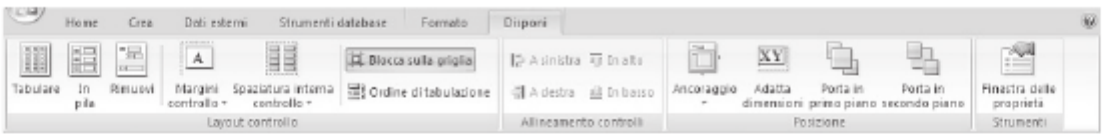


Figura 8.22 *La barra degli strumenti Disponi (visualizzazione layout)*

Figura 8.23 *La barra degli strumenti Disponi (visualizzazione struttura)*

Gli strumenti *Intestazione/Pié di pagina maschera* e *Intestazione/Pié di pagina pagina* del gruppo *Mostra/Nascondi* della barra ***Disponi*** permettono di visualizzare o nascondere **intestazioni** e **piè di pagina**; un piè di pagina è sempre abbinato a una intestazione corrispondente. Intestazione e piè di pagina maschera possono contenere informazioni generali sulla maschera (come il titolo o istruzioni per l'uso della maschera) e pulsanti; intestazione e piè di pagina pagina permettono di inserire informazioni per la stampa.
Si può ridimensionare una **sezione** trascinando il bordo superiore della sezione successiva; si possono impostare le proprietà di una sezione nella finestra delle proprietà quando è selezionata la sezione; si può selezionare una sezione facendo clic sul quadratino a sinistra all'inizio della sezione.

Proprietà dei controlli

Si possono impostare le proprietà di un controllo nella ***Finestra delle proprietà*** a cui si accede, dopo aver selezionato il controllo, con lo strumento *Finestra delle proprietà* del gruppo *Strumenti* della barra *Struttura*.

Nota Le proprietà sono divise nelle schede *Formato* (caratteristiche di presentazione), *Dati* (origine dei valori), *Evento*, *Altro*; la scheda *Tutte* visualizza tutte le proprietà insieme.

Le proprietà disponibili dipendono dal tipo di controllo.

Nota Si possono selezionare più controlli e modificare le proprietà per tutto il gruppo; se si selezionano controlli di tipo diverso vengono visualizzate solo le proprietà comuni.

Alcune proprietà che determinano l'aspetto generale e il funzionamento di ciascun tipo di controllo sono predefinite; si possono modificare le **impostazioni predefinite** di un controllo impostandole nella finestra delle proprietà quando è selezionato lo strumento corrispondente nel gruppo *Controlli* della barra *Struttura*.

Proprietà dei controlli associati

Un **controllo associato** deriva le impostazioni di alcune proprietà dal campo corrispondente della tabella (specificato nella proprietà ***Origine controllo***); per esempio vengono mantenuti i valori assegnati alle proprietà *Formato* e *Posizioni decimali*; il valore della proprietà *Testo barra stato* è la descrizione associata al campi della tabella nella griglia di definizione dei campi.

Le modifiche apportate alle proprietà dei controlli sulla maschera non hanno effetto sui campi della tabella di origine; analogamente non hanno effetto sulla maschera le modifiche apportate alle proprietà dei campi della tabella, dopo che la maschera è stata creata.

Loescher Editore - Vietata la vendita e la diffusione

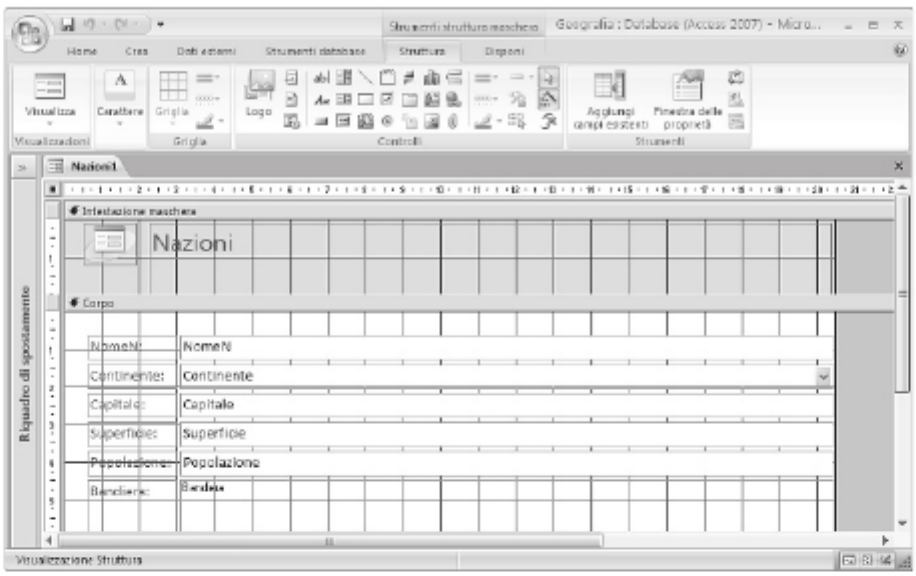


Figura 8.24 *Maschera basata sulla tabella Nazioni (creata con lo strumento Maschera) in visualizzazione struttura*

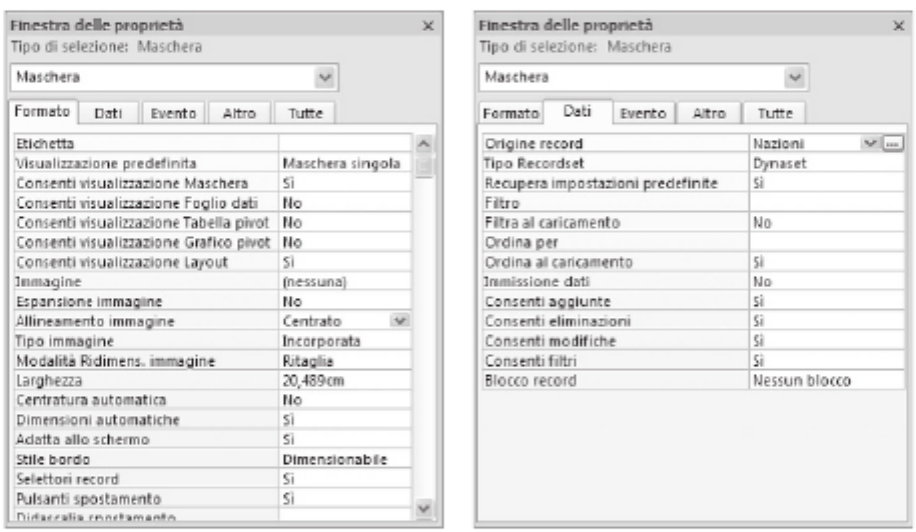


Figura 8.25 *Schede Formato e Dati delle proprietà della maschera della figura 8.24*

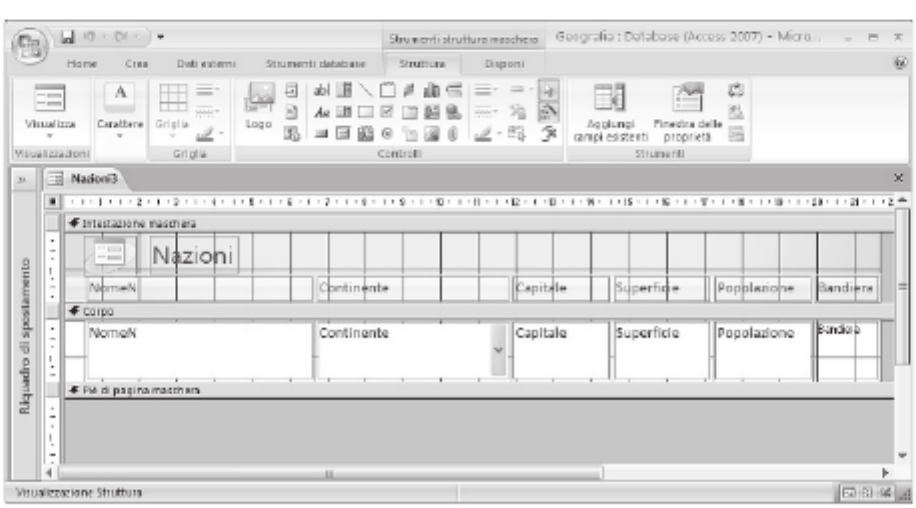


Figura 8.26 *Maschera basata sulla tabella Nazioni (creata con lo strumento Più elementi) in visualizzazione struttura*

Loescher Editore - Vietata la vendita e la diffusione

Controlli e Proprietà

Etichette

Dopo aver selezionato lo strumento ***Etichetta*** basta fare clic nel punto desiderato della maschera e digitare il testo; le dimensioni del controllo si adattano automaticamente al testo digitato; è possibile anche trascinare il puntatore per creare un controllo delle dimensioni desiderate, prima di digitare il testo.

Un'etichetta creata in questo modo non è allegata a nessun controllo; si può associare in seguito l'etichetta a un controllo. **Nota**

Caselle di testo calcolate

Una **casella di testo calcolata** è un controllo non associato: non si riferisce ai dati di un campo di una tabella, ma visualizza il risultato di un'espressione.

Dopo aver creato il controllo con lo strumento *Casella di testo*, si può scrivere l'espressione da calcolare preceduta dal segno di uguale (=) direttamente nella casella o nella proprietà ***Origine controllo*** della scheda *Dati*.

Per scrivere l'espressione nella proprietà *Origine controllo* è possibile richiamare il *Generatore di espressioni* facendo clic sul pulsante con i tre puntini. **Nota**

Se l'espressione coinvolge un campo non presente nella maschera si verifica un errore: è necessario aggiungere il campo (o la tabella che lo contiene) nella proprietà *Origine record* della maschera. **Nota**

Creazione di controlli associati e non associati

Per creare un **controllo associato**, in alternativa:

- selezionare lo strumento e trascinare il puntatore sulla maschera quando lo strumento *Usa creazioni guidate controllo* è abilitato; viene avviata una procedura guidata per scegliere il campo da associare;
- trascinare il campo dalla finestra *Elenco campi* dopo aver selezionato lo strumento;
- impostare la proprietà *Origine controllo* del controllo dopo averlo creato.

Per creare un **controllo non associato**:

- selezionare lo strumento e trascinare il puntatore sulla maschera quando lo strumento *Usa creazioni guidate controllo* è disabilitato.

Anche la procedura guidata comunque permette la creazione di un controllo non associato. **Nota**

Caselle di riepilogo e Caselle combinate

Proprietà principali (scheda *Dati*):

Consenti modifiche a elenco valori il valore *No* impedisce la modifica delle voci presenti nell'elenco;
Solo in elenco il valore *Sì* impedisce l'immissione di valori diversi da quelli proposti nell'elenco;
Tipo origine riga indica se prendere i valori da un campo di una tabella o da un elenco definito appositamente;
Origine riga è il nome della tabella o l'elenco di valori.

Loescher Editore - Vietata la vendita e la diffusione

Per un elenco di valori fissi bisogna impostare *Tipo origine riga* a *Elenco valori* e digitare i valori che costituiscono l'elenco separati da punti e virgola; per semplificare l'inserimento si può usare la finestra *Modifica voci di elenco* che si apre facendo clic sul pulsante con i tre puntini.

Per un elenco di valori presi da una tabella si deve impostare *Tipo origine riga* a *Tabella/query* e *Origine riga* al nome della tabella; per default viene utilizzata la prima colonna della tabella; per usare una colonna diversa bisogna impostare le proprietà *Numero colonne* (della scheda *Altro*) e *Colonna associata* (della scheda *Dati*) e nascondere le colonne precedenti, indicando per ciascuna di esse una larghezza 0 nella proprietà *Larghezza colonne* (della scheda *Altro*), inserendo un punto e virgola tra uno zero e l'altro.

Gruppi di pulsanti di opzione

Conviene usare la procedura guidata avviata trascinando il mouse dopo aver selezionato lo strumento ***Gruppo di opzioni*** con lo strumento *Usa creazioni guidate controllo* attivo.
La procedura guidata permette di impostare le etichette di ciascuna opzione, il valore predefinito, i valori (numerici) corrispondenti alle etichette, l'eventuale campo associato, il tipo di controlli da raggruppare (oltre a pulsanti di opzione si potrebbero usare caselle di controllo e interruttori) e l'etichetta visualizzata nel bordo del gruppo di opzioni.

Pulsanti di comando

Un **pulsante di comando** permette di richiamare una macro quando viene selezionato.
La proprietà ***SuClic*** nella scheda *Evento* indica il nome della macro o della routine evento da eseguire quando si fa clic sul pulsante.

> La procedura guidata permette di creare automaticamente un pulsante di comando scegliendo l'azione da eseguire da un elenco suddiviso per categorie. **Nota**

Le proprietà *Etichetta* e *Immagine* permettono di definire il testo o l'immagine che devono comparire sul pulsante.

> Perché il testo della proprietà *Etichetta* sia visibile, la proprietà *Disposizione didascalia immagine* non deve essere impostata al valore *Nessuna didascalia*. **Nota**

Maschere con sottomaschere

Una maschera può contenere campi che provengono da più tabelle e può basarsi su query che combinano dati di più tabelle.
Una maschera con campi che provengono da più tabelle è detta **semplice** se visualizza campi di un solo record di ciascuna tabella, **complessa** se contiene una sottomaschera.

> Una maschera può contenere un numero qualsiasi di sottomaschere e possono esserci più livelli di sottomaschere. **Nota**

Una maschera con una **sottomaschera** permette di visualizzare i dati di tabelle legate da una relazione uno a molti; la maschera principale visualizza il record del lato uno, la sottomaschera visualizza i record correlati del lato molti.

Si possono creare la maschera e la sottomaschera separatamente, lasciando nella maschera principale lo spazio necessario alla sottomaschera. Dopo averla chiusa, si può trascinare la sottomaschera dal riquadro di spostamento alla maschera principale in visualizzazione struttura.

> Se è stabilita la relazione tra le tabelle lo strumento *Maschera* crea automaticamente una maschera con sottomaschere. **Nota**

Loescher Editore - Vietata la vendita e la diffusione

8.4 MASCHERE

Le **maschere** permettono di creare finestre con layout personalizzati per organizzare i dati nel modo desiderato.

Creazione di maschere

Per creare maschere si usano gli strumenti del gruppo *Maschere* della barra *Crea*.

Lo strumento ***Maschera*** crea automaticamente una maschera basata sulla tabella selezionata, con un'etichetta e una casella di testo per ogni campo della tabella disposti in colonna uno sotto l'altro (disposizione a colonne); possono essere creati controlli diversi da caselle di testo in base al tipo di dati dei campi e alle proprietà impostate.

Nota

Se sulla tabella sono definite delle relazioni vengono create automaticamente maschere complesse (vedi pagina 229).

La maschera è in ***Visualizzazione Layout***; sono visibili i dati ed è possibile modificare la posizione e l'aspetto dei controlli mediante le barre *Disponi e Formato* (in primo piano).
Lo strumento *Visualizza* del gruppo *Visualizzazioni* della barra *Formato* permette di passare alle modalità ***Visualizzazione Struttura*** (con la barra *Struttura*) in cui è possibile modificare la struttura della maschera cambiando numero e tipo di controlli, alla modalità ***Visualizzazione Maschera*** in cui si usa la maschera per accedere ai dati (scorrere i record, modificarli e aggiungerne di nuovi) e alla ***Visualizzazione Foglio dati*** in cui si ottiene il foglio dati della tabella su cui è basata la maschera.

Altri strumenti del gruppo *Maschere* sono:

Maschera divisa mostra sulla stessa scheda la maschera e il foglio dati; in visualizzazione maschera si può operare sui dati sia dalla maschera che dal foglio dati;

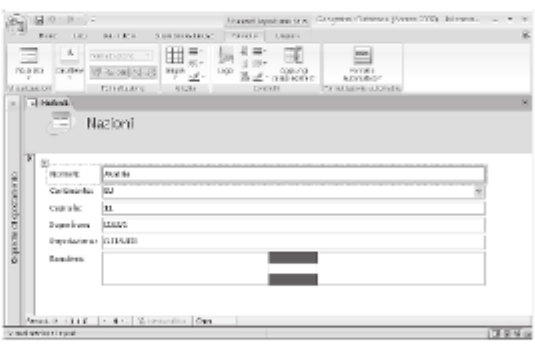


Figura 8.13 *Maschera creata con: Maschera*

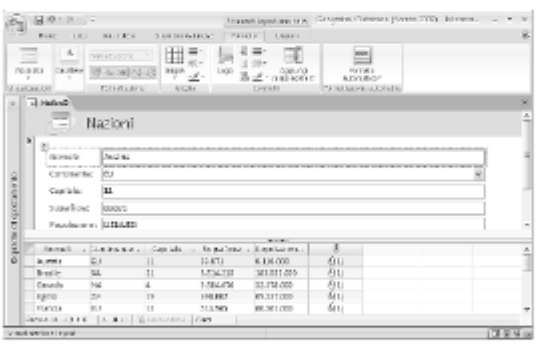


Figura 8.14 *Maschera creata con: Maschera divisa*

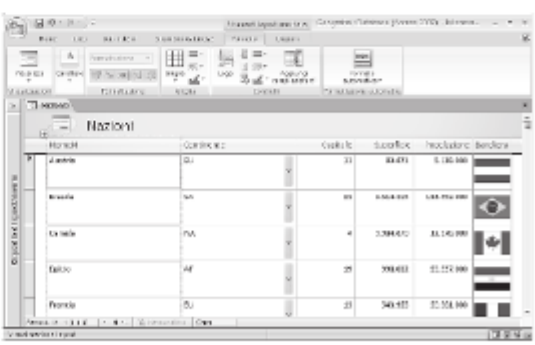


Figura 8.15 *Maschera creata con: Più elementi*

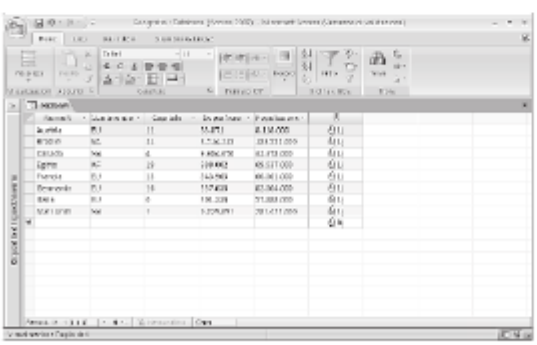
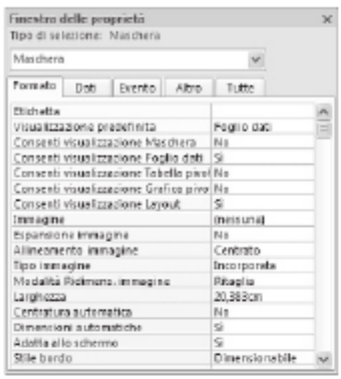


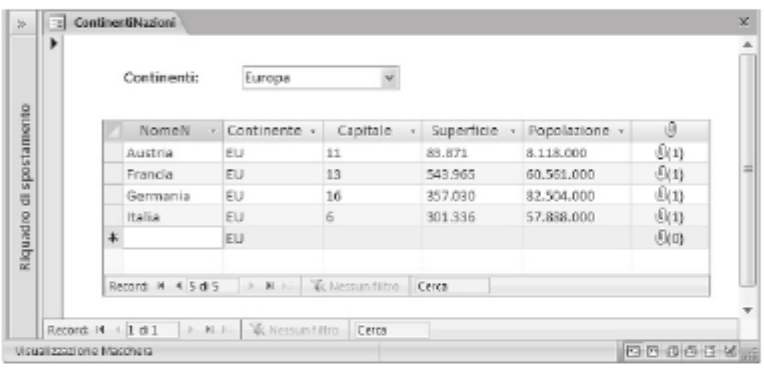
Figura 8.16 *Maschera creata con: Foglio dati*

Loescher Editore - Vietata la vendita e la diffusione



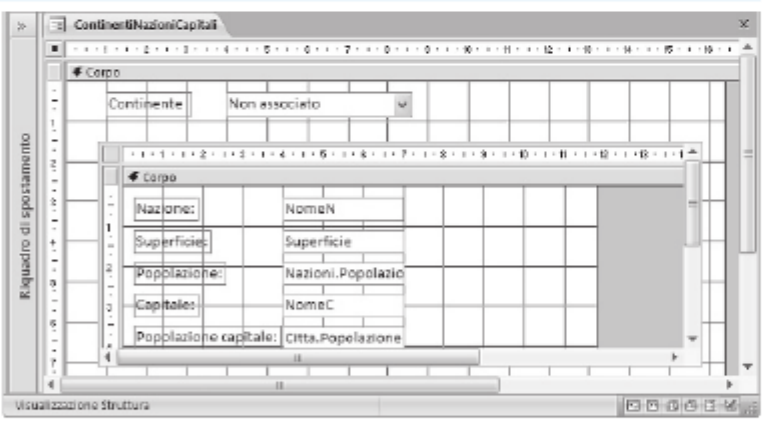
La sottomaschera ha come visualizzazione predefinita foglio dati.

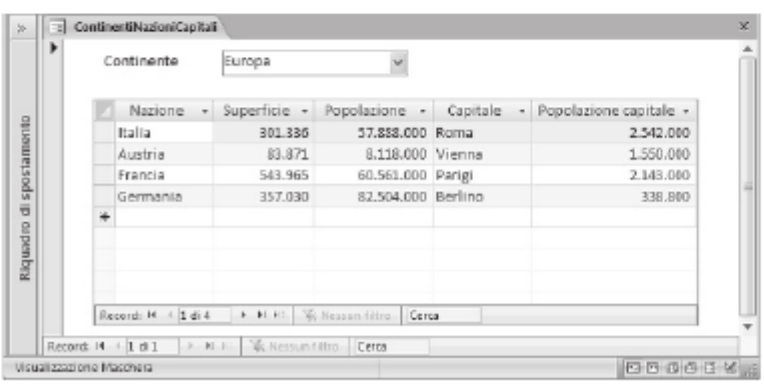
In visualizzazione maschera. selezionato un continente, si vedono le nazioni del continente ed è possibile inserire e modificare i dati.



Esempio 8.19

Maschera principale col nome del continente e sottomaschera basata su una query che ricava i dati dalle tabelle *Nazioni* e *Città*.
La query deve includere il campo *Continente* di *Nazioni*.





Loescher Editore - Vietata la vendita e la diffusione

8.5 REPORT

Creazione di report

I **report** permettono di realizzare prospetti di stampa organizzando le informazioni nel modo desiderato, definendone la disposizione e l'aspetto e aggiungendo elementi di vario tipo come testo, dati, linee, caselle, immagini o grafici.
Si possono creare report con gli strumenti del gruppo *Report* della barra *Crea*.

Lo strumento più semplice è ***Report*** che consente di creare un report partendo da una tabella o una query selezionata nel riquadro di spostamento. Viene creato un report che presenta in forma tabellare i record dell'oggetto di partenza.

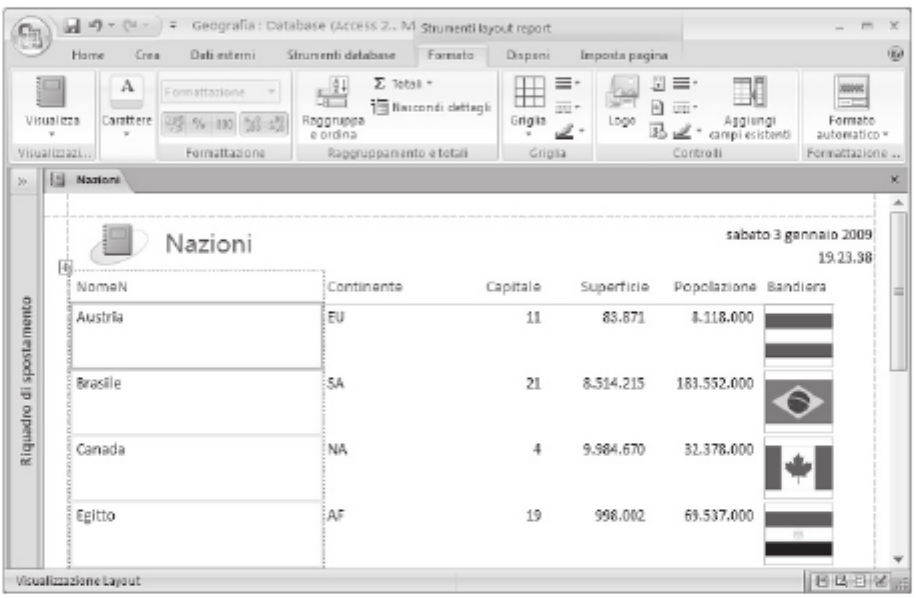


Figura 8.27
Report basato sulla tabella Nazioni creato con lo strumento Report (visualizzazione layout)

Il report è in ***Visualizzazione Layout***; si possono ridimensionare i controlli trascinandone i bordi e si può personalizzare il report con gli strumenti del gruppo *Carattere* della barra *Formato*. Lo strumento *Visualizza* del gruppo *Visualizzazioni Formato* della barre *Formato* o *Home* permette di passare alle modalità ***Visualizzazione struttura***, ***Visualizzazione report*** e ***Anteprima di stampa***.

Altri strumenti del gruppo *Report* sono:

Report vuoto crea un report vuoto in visualizzazione layout e apre la finestra *Elenco campi* da cui è possibile scegliere i campi da inserire nel report, trascinandoli; è possibile modificare la posizione e l'aspetto dei controlli;

Struttura report crea un report vuoto in visualizzazione struttura e apre la finestra *Elenco campi*;

Etichette avvia una procedura guidata per la creazione di un report (basato sull'oggetto selezionato nel riquadro di spostamento) adatto per la stampa su etichette; si possono stabilire le dimensioni delle etichette, l'aspetto e i campi da inserire.

Si può impostare o modificare la tabella o query su cui si basa il report con la proprietà ***Origine record*** del report (nella scheda *Dati*); si può creare o modificare una query richiamando il *Generatore query* facendo clic sul pulsante con i tre puntini.

Nota
Possono esserci anche report che non si basano su nessuna tabella (per esempio un report per la stampa di una lettera).

Loescher Editore - Vietata la vendita e la diffusione

Si può **stampare** il report con il comando *Stampa* dal pulsante *Office* da qualsiasi modalità di visualizzazione.

Gli strumenti del gruppo ***Esporta*** della barra *Dati esterni* consentono di salvare le informazioni del report in altri formati, tra cui RTF, XML e PDF (se è installato l'apposito componente aggiuntivo).

Nota

I componenti aggiuntivi possono essere installati con il comando *Salva con nome*, *Trova componenti aggiuntivi per altri formati di file* del pulsante *Office*.

CREAZIONE GUIDATA DI UN REPORT

Si può creare un report con una procedura guidata (strumento ***Creazione guidata Report*** del gruppo *Report* della barra *Crea*).
Si può scegliere su quali tabelle o query basare il report, e definire l'aspetto del report impostando anche eventuali calcoli e raggruppamenti.

I CONTROLLI

La struttura del report è definita mediante **controlli** associati o non associati ai campi.
I controlli possono essere aggiunti o modificati in visualizzazione struttura del report; per creare i controlli si utilizzano gli strumenti del gruppo *Controlli* della barra *Struttura*.

Nota

Per disporre più facilmente i controlli si possono usare griglie e righelli (strumenti *Griglia* e *Righello* del gruppo *Mostra/Nascondi*).

ATTENZIONE

I controlli inseriti nel report vengono ripetuti per ogni record da stampare; vanno quindi inseriti i controlli relativi alla stampa di **un solo record**.

Nota

È possibile usare nel report anche caselle di selezione e gruppi di pulsanti di opzioni che appaiono selezionati o no in base al valore da rappresentare.

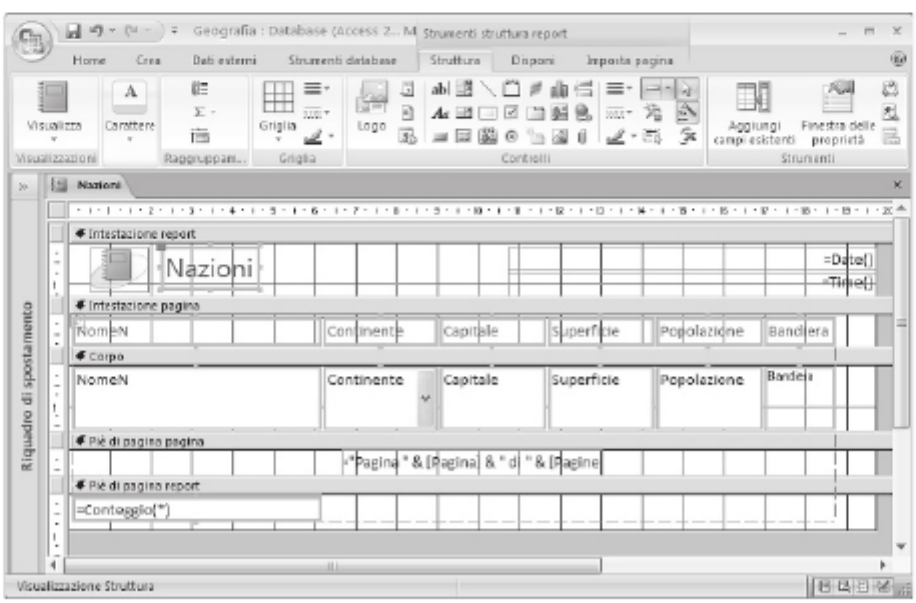


Figura 8.28 *Report basato sulla tabella Nazioni (creato con lo strumento Report)in visualizzazione struttura*

Loescher Editore - Vietata la vendita e la diffusione

Le proprietà dei controlli possono essere impostate e modificate nella finestra delle proprietà. Alcune proprietà utili sono:

Espandibile il valore *Sì* fa ingrandire il controllo per rendere visibile tutto il testo contenuto;
Riducibile il valore *Sì* non fa stampare il controllo se contiene valori nulli;
Nascondi duplicati il valore *Sì* non fa stampare il controllo se contiene un valore uguale a quello del record precedente.

Le sezioni

Gli strumenti *Intestazione/Pié di pagina report* e *Intestazione/Pié di pagina pagina* del gruppo *Mostra/Nascondi* della barra ***Disponi*** permettono di visualizzare o nascondere **intestazioni** e **piè di pagina**; un piè di pagina è sempre abbinato a una intestazione corrispondente.

Nota

Si può ridimensionare una sezione trascinando il bordo superiore della sezione successiva; si possono impostare le proprietà di una sezione nella finestra delle proprietà quando è selezionata la sezione; si può selezionare una sezione facendo clic sul quadratino a sinistra all'inizio della sezione.

Intestazione e piè di pagina **report** possono contenere informazioni generali e vengono stampati una sola volta, rispettivamente all'inizio e alla fine del report; l'intestazione di solito contiene il titolo e un eventuale logo, il piè di pagina può essere usato per stampare totali del report.

Intestazione e piè di pagina **pagina** vengono stampati per ogni pagina; l'intestazione può essere usata per stampare le intestazioni di colonna; intestazione o piè di pagina possono contenere informazioni da riportare su ogni pagina come la data o il numero di pagina.

Attenzione

L'intestazione report viene stampata sulla prima pagina prima dell'eventuale intestazione di pagina; il piè di pagina report viene stampato sull'ultima pagina prima dell'eventuale piè di pagina della pagina.

Nota

Le proprietà *Intestazione (pagina)* e *Piè di pagina (pagina)* del report permettono di specificare se l'intestazione e il piè di pagina pagina devono essere stampati anche nella prima e nell'ultima pagina insieme all'intestazione e al piè di pagina report.

Nel gruppo *Controlli* della barra *Struttura* sono disponibili anche gli strumenti:

Logo per inserire un'immagine nell'intestazione di report;
Titolo per inserire un testo (etichetta) nell'intestazione di report;
Inserisci numero di pagina permette di scegliere la posizione (nell'intestazione o nel piè di pagina pagina) e il formato (numero o numero/totale) del numero di pagina;
Data e ora permette di inserire data e ora correnti, scegliendone il formato;
Inserisci o rimuovi interruzioni di pagina permette di inserire interruzioni di pagina per far iniziare una sezione su una nuova pagina o suddividere una sezione su più pagine.

Ordinamento e raggruppamento di dati

Si possono organizzare i dati da stampare in un report ordinandoli in base a uno o più campi e raggruppando i record che hanno lo stesso valore in un campo, o una parte del campo uguale.

Loescher Editore - Vietata la vendita e la diffusione

Quando si creano **raggruppamenti** di record si possono inserire nel report una **intestazione** e un **piè di pagina gruppo** per ogni gruppo di record; l'intestazione in genere contiene informazioni introduttive (come il nome del gruppo), il piè di pagina informazioni di riepilogo (come i totali relativi al gruppo).

Nota

Automaticamente viene inserita l'intestazione di gruppo.

In visualizzazione struttura lo strumento ***Raggruppa e ordina*** del gruppo *Raggruppamento e totali* della barra *Struttura* apre il riquadro *Raggruppamento, ordinamento e totale* in cui sono presenti i pulsanti *Aggiungi gruppo* e *Aggiungi ordinamento*; facendo clic sui pulsanti si apre un elenco da cui scegliere i campi su cui effettuare ordinamenti e raggruppamenti.
Viene presentata una barra che riporta l'operazione (ordinamento o raggruppamento) e permette di impostare varie opzioni; per il raggruppamento si può scegliere il criterio di raggruppamento, i calcoli da effettuare alla fine del gruppo, la posizione in cui visualizzare i risultati (intestazione o piè di pagina del gruppo), la presenza dell'intestazione e del piè di pagina, le interruzioni di pagina rispetto ai gruppi.

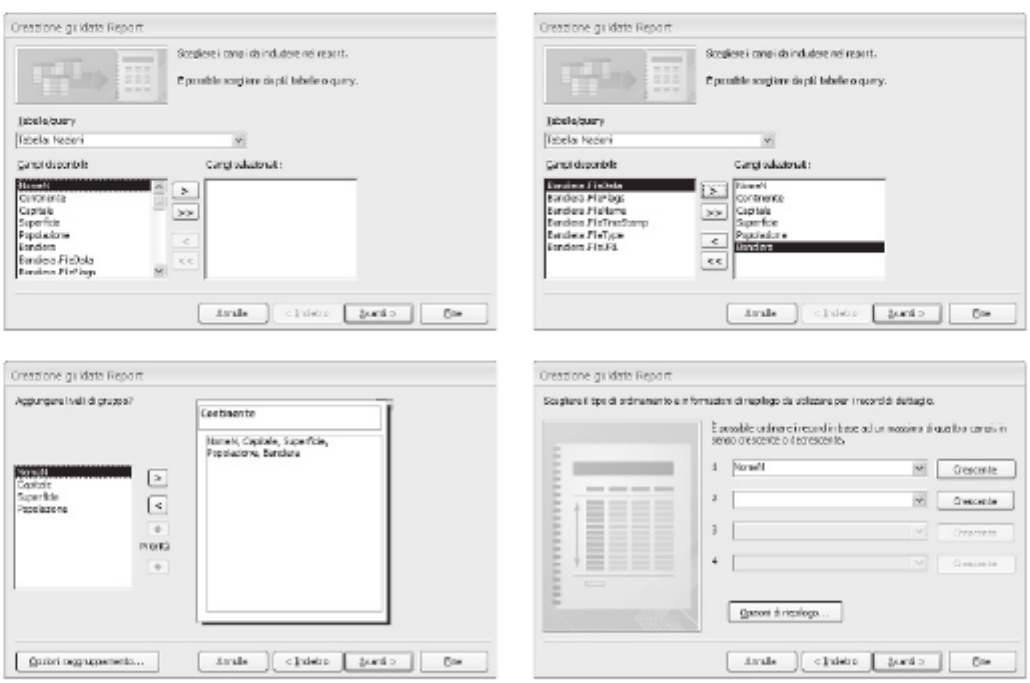
I campi di tipo *Testo* possono essere raggruppati anche in base ai primi caratteri del campo; i campi di tipo *Numerico*, *Data/ora*, *Contatore* e *Valuta* possono essere raggruppati anche in base a intervalli di valori.

Lo strumento ***Totali*** del gruppo *Raggruppamento e totali* della barra *Struttura* inserisce automaticamente un controllo calcolato (casella di testo contenente la formula scelta) nel piè di pagina gruppo e nel piè di pagina report.

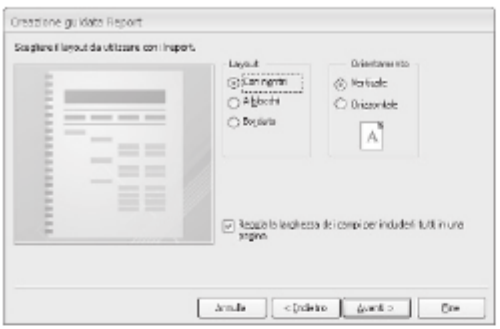
Esempio 8.20

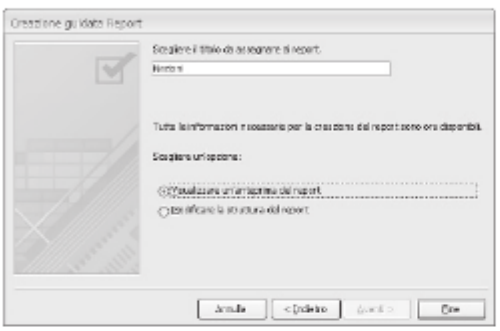
Per ogni continente elenco delle nazioni e alla fine del gruppo numero di nazioni, somma delle superficie e media della popolazione. Alla fine del report superficie totale e media della popolazione.

Conviene usare la creazione guidata:

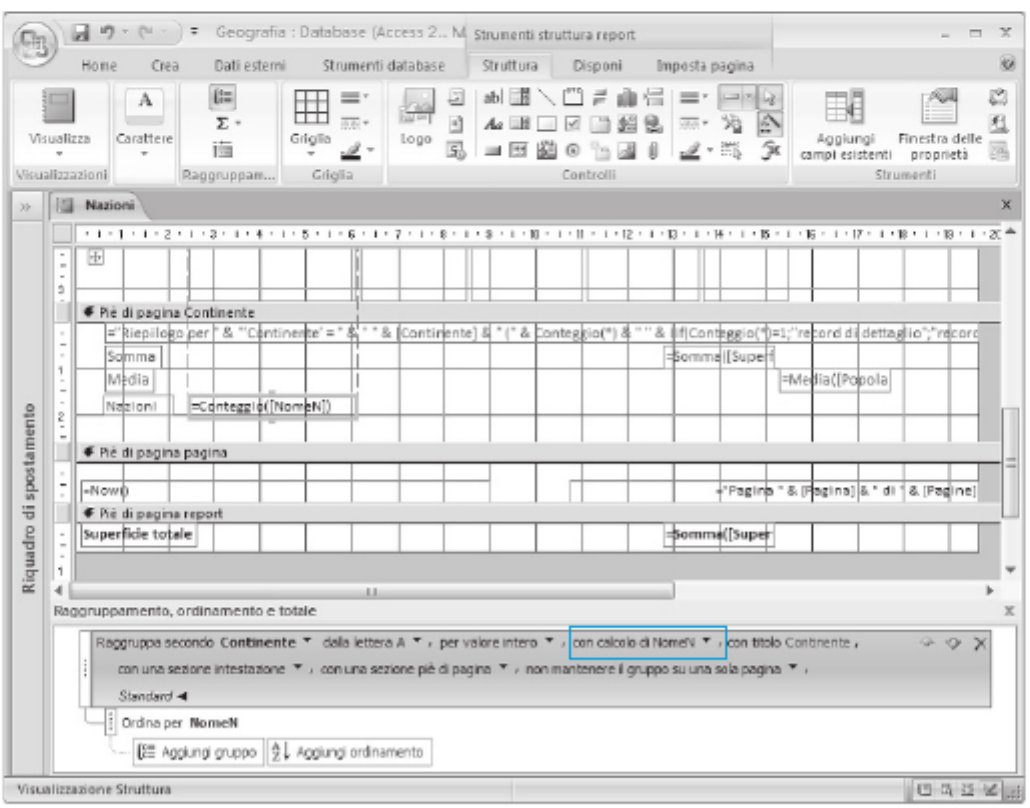


Facendo clic sul pulsante *Opzioni di riepilogo...*





La creazione guidata non permette di specificare il conteggio delle nazioni per continente; è necessario passare in visualizzazione struttura e impostarlo nella barra di raggruppamento del campo *Continente*. Viene aggiunta una casella di testo calcolata nel piè di pagina gruppo: basta posizionarla correttamente e aggiungere un'etichetta esplicativa.



Loescher Editore - Vietata la vendita e la diffusione

Sottoreport

Un **sottoreport** è un report inserito all'interno di un altro report.
Il report principale può non essere associato ad alcuna tabella o query e servire solo per combinare insieme due o più sottoreport non correlati.
Si può usare un report con un sottoreport per stampare i dati di tabelle legate da una relazione uno a molti; in questo caso il sottoreport contiene dati correlati al report principale.

Si possono creare il report e il sottoreport separatamente; dopo averlo chiuso, si può trascinare il sottoreport dal riquadro di spostamento al report principale in visualizzazione struttura.
Dopo aver inserito il sottoreport nel report principale, lo si può modificare dalla visualizzazione struttura del report principale.

Passando alla visualizzazione report del report principale si ottiene il report con il corrispondente sottoreport.

Se tra le tabelle è impostata una relazione, il sottoreport viene collegato automaticamente al report principale in modo da visualizzare soltanto i record correlati.

Il collegamento tra report principale e sottoreport avviene tramite le proprietà ***Collega campi master*** e ***Collega campi secondari*** del sottoreport (selezionato in visualizzazione struttura del report principale).
Collega campi master viene indica il nome del campo di collegamento della tabella dal lato uno della relazione; *Collega campi secondari* indica il nome del campo della tabella dal lato molti.

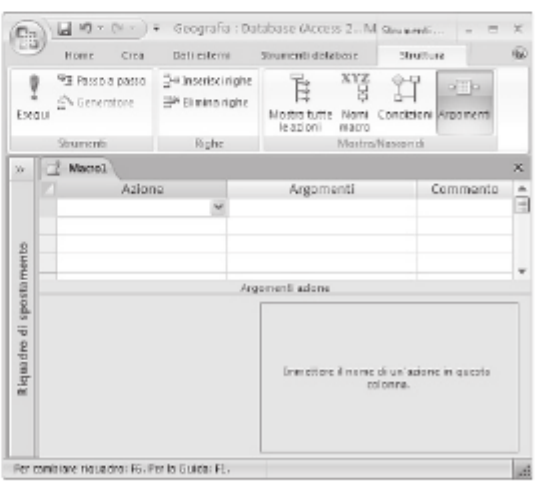
8.6 MACRO

Creazione ed esecuzione di una macro

Le **macro** permettono di eseguire automaticamente operazioni (**azioni**) sugli oggetti del database.
Per **creare** una macro si usa lo strumento ***Macro*** del gruppo *Altro* della barra *Crea*.
Si apre una scheda con una griglia per la definizione delle azioni e il riquadro *Argomenti azione* per la scrittura dei parametri dell'azione scelta.
Si può richiedere l'**esecuzione** della macro con lo strumento ***Esegui*** del gruppo *Strumenti* della barra *Struttura*.

Nota

Prima di eseguire una macro bisogna salvarla.



Si può eseguire una macro anche facendo clic destro sul nome della macro nel riquadro di spostamento e scegliendo *Esegui* dal menu di scelta rapida.

Si può **modificare** una macro esistente aprendola in modalità visualizzazione struttura (facendo clic destro sul nome della macro nel riquadro di spostamento e scegliendo *Visualizzazione Struttura* dal menu di scelta rapida).

Figura 8.29
Definizione di una macro

Loescher Editore - Vietata la vendita e la diffusione

La griglia di definizione delle azioni presenta le colonne:

Azione	nome dell'azione; può essere selezionato da una casella di riepilogo con l'elenco delle azioni disponibili;
Argomenti	parametri necessari all'azione, presi dal riquadro *Argomenti azione*; la colonna può essere nascosta disattivando lo strumento *Argomenti* del gruppo *Mostra/Nascondi* della barra *Struttura*;
Commento	descrizione dell'operazione eseguita dalla macro.

Lo strumento *Condizioni* del gruppo *Mostra/Nascondi* della barra *Struttura* aggiunge la colonna ***Condizione*** in cui è possibile specificare una condizione che deve essere vera perché l'azione corrispondente sia eseguita; per la scrittura della condizione si può utilizzare il *Generatore di espressioni* attivabile con lo strumento *Generatore* del gruppo *Strumenti*.
Per condizionare una serie di azioni si devono digitare dei puntini di sospensione (...) nella colonna *Condizione* delle azioni successive; se la condizione è falsa viene eseguita la prima azione in cui non sono presenti i puntini di sospensione nella colonna *Condizione*.

Azioni utilizzabili nelle macro

Le **azioni** selezionabili nella colonna *Azione* permettono di eseguire operazioni di vario tipo come:

apertura e chiusura di oggetti	*ApriMaschera*, *ApriTabella*, *ApriQuery*, *ApriReport*, *Chiudi*;
spostamento tra controlli, record o pagine	*VaiAControllo*, *VaiARecord, VaiAPagina*;
ricerca di record	*TrovaRecord*, *TrovaSuccessivo*;
applicazione di filtri	*ApplicaFiltro* (per specificare i record da visualizzare bisogna inserire un'espressione nell'argomento *Condizione Where* o il nome di una query nell'argomento *Nome filtro*);
esecuzione di un'altra macro	*EseguiMacro*;
visualizzazione di un messaggio	*FinestraMessaggio*;
annullamento di un evento	*AnnullaEvento*.

Le azioni che aprono un oggetto del database possono essere impostate automaticamente trascinando l'oggetto dal riquadro di spostamento in una casella della colonna *Azione*.
Se un argomento di un'azione è costituito da un oggetto, è possibile impostarlo trascinando l'oggetto sulla casella relativa all'argomento nel riquadro *Argomenti azione*.
Il trascinamento di una macro dal riquadro di spostamento in una casella della colonna *Azione* crea un'azione *EseguiMacro* (permette di richiamare l'esecuzione di un'altra macro).
L'azione *FinestraMessaggio* permette di visualizzare una finestra di dialogo predefinita; il testo del messaggio da visualizzare e l'eventuale titolo della finestra devono essere specificati nelle proprietà *Messaggio* e *Titolo* del riquadro *Argomenti azione*.

Azioni non attendibili

Le azioni possibili in un database sono divise in **azioni attendibili** e **azioni non attendibili**.
Le azioni selezionabili nella colonna *Azione* sono attendibili. È possibile selezionare anche le azioni non attendibili attivando lo strumento *Mostra tutte le azioni* del gruppo *Mostra/Nascondi*.
Alcune azioni non attendibili sono *CopiaOggetto*, *EliminaOggetto*, *EseguiSQL*, *EseguiApplicazione*, *Stampa*, *TrasferisciDatabase*.

Loescher Editore - Vietata la vendita e la diffusione

Selezionando una azione non attendibile, nel selettore di riga dell'azione compare un'icona di pericolo (triangolino giallo) e in esecuzione la macro potrebbe bloccarsi se il livello di protezione è impostato al blocco del codice ritenuto pericoloso; in tal caso compare una finestra informativa e poi la finestra *Azione non riuscita* per arrestare la macro.

I livelli di protezione relativi alle macro possono essere impostati nella finestra ***Centro di protezione*** facendo clic sul pulsante *Impostazione Centro protezione...* e selezionando *Impostazioni macro*. Per aprire la finestra *Centro di protezione* si deve fare clic sul pulsante *Opzioni di Access* dalla finestra del pulsante *Office* e scegliere *Centro di protezione*.

Figura 8.30 *La finestra Centro di protezione*

Esecuzione passo a passo

Per verificare se una macro funziona correttamente è possibile richiedere l'esecuzione della macro azione per azione con lo strumento ***Passo a passo*** del gruppo *Strumenti*.

Nota Lo strumento deve essere attivato prima di richiedere l'esecuzione della macro.

Durante l'esecuzione della macro viene visualizzata la finestra ***Macro passo a passo*** che visualizza il nome della macro, il nome dell'azione da eseguire e gli argomenti dell'azione.

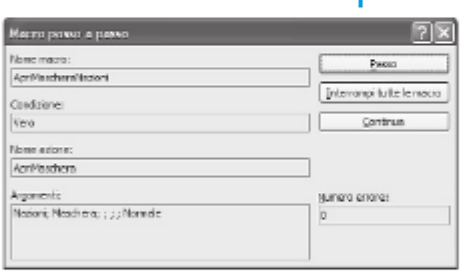


Figura 8.31 *La finestra Macro passo a passo*

La finestra *Macro passo a passo* presenta i pulsanti:

Passo	permette di eseguire l'azione e passare alle informazioni relative all'azione successiva.
Interrompi tutte le macro	permette di arrestare l'esecuzione della macro;
Continua	permette di eseguire il resto della macro disattivando la modalità passo a passo.

Se si verifica un errore il nome della finestra viene modificato in *Azione non riuscita* con attivo il solo pulsante *Interrompi tutte le macro*.

Nota Nella finestra è segnalato un numero di errore di riferimento per la ricerca di informazioni sul sito di Microsoft.

Gruppi macro

Può risultare utile raggruppare più macro in un **gruppo macro**, in modo da poterle visualizzare in un'unica scheda.
Si deve creare una macro col nome del gruppo.
Bisogna visualizzare anche la colonna ***Nome macro*** con lo strumento *Nomi macro* del gruppo *Mostra/Nascondi* della barra *Struttura*.

Loescher Editore - Vietata la vendita e la diffusione

Nella colonna *Nome macro* va inserito il nome di ciascuna macro, sulla riga relativa alla prima azione; la definizione della macro continua fino al nome della macro successiva.

Nazioni

Nome macro	Azione	Argomenti
ApriTabNazioni	ApriTabella	Nazioni; Foglio dati; Modifica
	VaiARecord	Tabella; Nazioni; Ultimo;
ApriMasNazioni	ApriMaschera	Nazioni; Maschera; ; ; Normale
	TrovaRecord	Italia; Campo intero; No; Tutto; No; Sì; Sì
StampaRepNazioni	ApriReport	Nazioni; Anteprima di stampa; ; ; Normale
	FinestraMessaggio	Report pronto per la stampa; Sì; Informazione;

Figura 8.32 *Definizione di un gruppo macro*

Nel riquadro di spostamento compare solo il nome del gruppo, ma ogni macro del gruppo ha un proprio nome e può essere richiamata singolarmente.
Per identificare una macro inclusa in un gruppo si usa la notazione *NomeGruppo.NomeMacro*.
Per **eseguire** una macro del gruppo si usa lo strumento *Esegui macro* del gruppo *Macro* della barra *Strumenti database*, dopo aver selezionato il gruppo nel riquadro di spostamento; una finestra di dialogo permette la scelta della macro.

Tasti di scelta rapida

Si possono assegnare azioni a **combinazioni di tasti** creando un gruppo macro con il nome ***AutoKeys***; ogni macro del gruppo ha come nome la combinazione di tasti (come azione si può usare *EseguiMacro*).

AutoKeys

Nome macro	Azione	Argomenti
^+T	EseguiMacro	Nazioni.ApriTabNazioni; ;
^+M	EseguiMacro	Nazioni.ApriMasNazioni; ;
^+R	EseguiMacro	Nazioni.StampaRepNazioni; ;

Figura 8.33 *Un gruppo macro Autokeys*

I nomi per le combinazioni di tasti seguono le convenzioni:
^ *Ctrl*; + *Shift*; {ins} *Ins*, {canc} *Canc*, {f1} *F1*, {f2} *F2* ecc.
Le combinazioni di tasti sono attive in qualsiasi punto del programma.

Gli eventi

Eventi

Un **evento** è un'azione riconosciuta da un oggetto; è un'azione **asincrona**, cioè che può accadere in qualsiasi momento, generata dal sistema o dall'interazione dell'utente col programma.

Ogni oggetto di Access riconosce una serie di eventi ed esegue in risposta delle operazioni predefinite.

Si possono suddividere gli **eventi** nelle seguenti categorie:

eventi di finestra	si verificano quando viene eseguita un'operazione su una maschera o un report, come l'apertura, la chiusura o il ridimensionamento;
eventi di attivazione	si verificano quando gli oggetti vengono attivati o disattivati, selezionati o deselezionati;
eventi di dati	si verificano quando vengono modificati dei dati;
eventi di tastiera	si verificano quando si digita qualcosa;
eventi di mouse	si verificano quando viene eseguita un'operazione con il mouse;
eventi di filtro	si verificano quando un filtro viene creato o applicato a una maschera;
eventi di stampa	si verificano quando un report viene formattato o stampato;
eventi di errore	si verificano quando viene riscontrato un errore;
eventi di timer	si verificano quando è trascorso un determinato intervallo di tempo.

Loescher Editore - Vietata la vendita e la diffusione

Ogni operazione genera una sequenza di eventi; individuando l'evento corretto è possibile eseguire una operazione esattamente nel momento in cui è necessaria.

Le maschere, i report e i relativi controlli dispongono di una **proprietà di evento** per ogni evento che può essere riconosciuto (le proprietà di evento sono elencate nella scheda ***Evento*** della finestra delle proprietà).
Si può personalizzare la risposta di un oggetto a un evento impostando la proprietà di evento relativa al nome di una macro o al valore *[Routine evento]*.

È possibile impostare la proprietà di evento al nome di una macro esistente (anche selezionandola da un elenco), o definire la macro al momento richiamando il *Generatore macro* (con il pulsante con i tre puntini a destra).

Si può usare il *Generatore macro* anche per modificare la macro. **Nota**

Se la proprietà di evento non è stata impostata viene eseguita solo la risposta automatica all'evento, se la proprietà di evento contiene il nome di una macro viene eseguita la macro, se contiene il valore *[Routine evento]* viene eseguita una routine evento scritta in Visual Basic.

Le routine evento in Visual Basic sono spiegate nel paragrafo 8.7. **Nota**

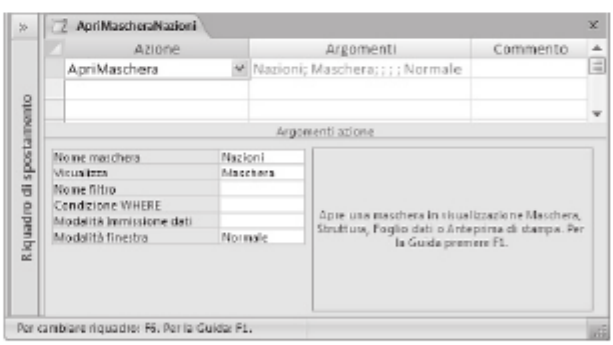
Annullamento di un evento

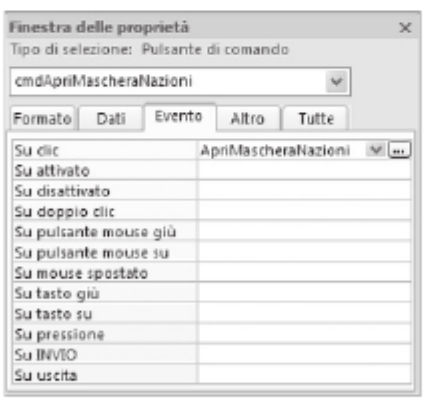
Si può **annullare** un evento usando in una macro l'azione ***AnnullaEvento***, condizionata per annullare l'evento quando si verificano particolari condizioni.

Non tutti gli eventi si possono annullare. **Nota**

Esempio 8.21

Pulsante di comando per l'apertura della maschera *Nazioni*. Bisogna definire la macro, creare una maschera con un pulsante e assegnare alla proprietà di evento *Su clic* del pulsante il nome della macro.





8.7 CODICE VBA

VBA

In Access è possibile scrivere codice in **VBA** (Visual Basic for Applications), un Visual Basic ottimizzato per operare sugli oggetti del database.

Loescher Editore - Vietata la vendita e la diffusione

Routine evento

Le **routine evento** sono routine Visual Basic eseguite al verificarsi di un evento. La proprietà di evento dell'oggetto deve essere impostata a ***[Routine evento]***.

Si può definire la routine attivando il ***Generatore di codice*** con il pulsante con i tre puntini a destra della casella relativa alla proprietà di evento.
L'**editor** di Visual Basic riporta nella barra del titolo il nome del database e l'oggetto a cui si riferisce il codice.

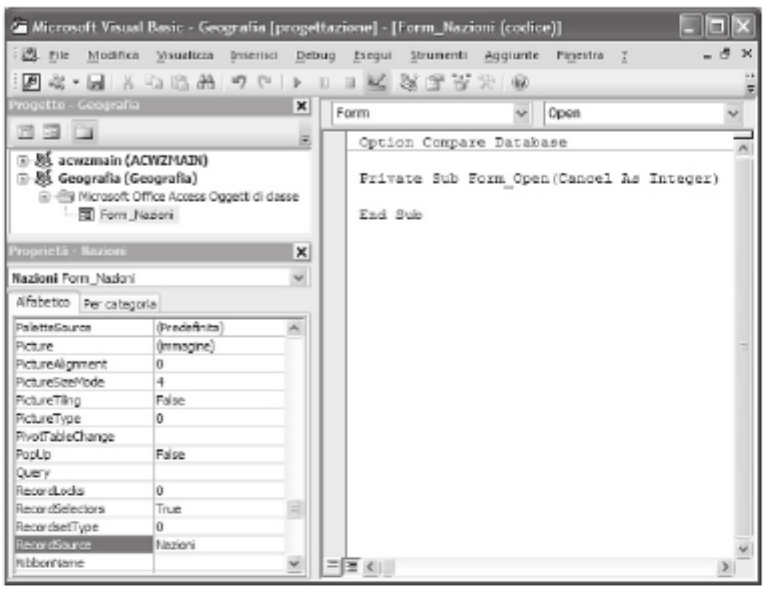


Figura 8.34
Finestra del codice per l'evento Su apertura della maschera Nazioni

Viene proposto automaticamente un modello per la routine evento; il **nome della routine** viene individuato in base all'oggetto e all'evento:

Form_*NomeEvento*	per una maschera
Report_*NomeEvento*	per un report
NomeControllo_NomeEvento	per un controllo

Si trova inserita anche l'istruzione ***Option compare database*** che stabilisce di usare il criterio impostato nelle opzioni di Access per il confronto tra stringhe.

Nota

Il criterio di confronto si imposta nella finestra *Opzioni di Access* (che si apre facendo clic sul pulsante *Opzioni di Access* della finestra del pulsante di *Office*); nella sezione *Creazione database* si può selezionare l'impostazione dalla casella di riepilogo *Ordinamento nuovi database*; il valore di default è *Generale* che non prevede distinzione tra caratteri minuscoli e maiuscoli.

Se una proprietà di evento è già impostata a una routine evento il *Generatore di codice* la apre nella finestra di codice permettendo di modificarla.
Il *Generatore di codice* si può richiamare anche scegliendo *Genera evento...* dal menu di scelta rapida che si ottiene facendo clic destro su una maschera o un controllo in visualizzazione struttura.

Confronto tra l'uso di macro e di routine evento

Molte operazioni possono essere gestite sia tramite macro che tramite routine Visual Basic.
I vantaggi dell'uso di **routine Visual Basic** sono:

- moduli incorporati nelle maschere e nei report e quindi facilità di gestione (per esempio per copiare una maschera);
- possibilità di definire funzioni personalizzate;
- possibilità di passare argomenti al codice durante l'esecuzione; nelle macro gli argomenti vengono impostati al momento della creazione della macro e non possono essere modificati durante l'esecuzione;

Loescher Editore - Vietata la vendita e la diffusione

- gestione degli errori con visualizzazione di messaggi personalizzati;
- possibilità di creare e modificare oggetti all'interno del codice;
- possibilità di eseguire operazioni su un record alla volta in un set di record.

Anche le **macro** però hanno alcuni vantaggi:

- alcune operazioni possono essere eseguite solo con le macro perché non esistono istruzioni Viual Basic equivalenti;
- solo le macro possono essere assegnate ai tasti di scelta rapida;
- le macro sono molto più sicure: se gli argomenti sono corretti, non possono danneggiare il database, mentre il codice Visual Basic non può essere limitato in alcun modo.

Per usare una routine quando è indispensabile una macro si può richiamare la routine nella macro con l'azione ***EseguiCodice***.

Per eseguire azioni macro nel codice di una routine Visual Basic si usano i metodi dell'oggetto ***DoCmd***.
La maggior parte delle azioni hanno un metodo di *DoCmd* corrispondente, che si richiama con la sintassi:

```
DoCmd.nomeAzione listaArgomenti
```

Gli argomenti in *listaArgomenti* vanno separati da virgole; se un argomento opzionale viene omesso, bisogna inserire un'ulteriore virgola.

```
DoCmd.OpenTable "Nazioni", , acReadOnly
```
Apre la tabella *Nazioni* senza permettere modifiche e inserimenti.

Nota

Per annullare un evento in una routine si richiama l'azione *AnnullaEvento* con il metodo *CancelEvent* dell'oggetto *DoCmd*.

I MODULI

I **moduli** sono oggetti del database che contengono codice Visual Basic (dichiarazioni e routine).
Per la scrittura di codice si deve aprire l'**editor** di Visual Basic con lo strumento ***Visual Basic*** del gruppo *Macro* della barra *Strumenti database*.
Nell'editor si può richiedere la **compilazione** del codice con il comando *Compila* (seguito dal nome del database) del menu *Debug*.

MODULI DI MASCHERA E DI REPORT

Le routine evento scritte per una maschera o un report e per i relativi controlli sono raccolte in un modulo associato alla maschera o al report (chiamato **modulo di maschera** o di **report**).
Il modulo ha come **nome** il nome della maschera o del report preceduto dal termine ***Form_*** per la maschera e ***Report_*** per il report.

Nota

I moduli di maschera e report possono contenere anche routine generali, non associate agli eventi.

Si può aprire il modulo di una maschera o di un report con lo strumento ***Visualizzazione Codice*** del gruppo *Strumenti* della barra *Struttura* quando la maschera o il report sono in modalità struttura.
Nell'editor si può selezionare una maschera o report nel riquadro ***Progetto***, espandendo la voce con il nome del database e poi *Microsoft Office Access Oggetti di classe*.

Loescher Editore - Vietata la vendita e la diffusione

Sopra al codice la casella a sinistra riporta l'**elenco dei controlli** della maschera, la casella a destra l'**elenco degli eventi** riconosciuti; i nomi degli eventi a cui sono associate routine evento sono in grassetto.

Moduli globali

I **moduli globali** sono indipendenti da maschere e report e contengono codice di uso generale richiamabile da altre routine di moduli globali, da routine evento, e da macro.

Si può creare un modulo con lo strumento ***Modulo*** del gruppo *Altro* della barra *Crea*;

Nota

Il nome dello strumento dipende dall'ultimo oggetto di questo tipo creato e diventa alternativamente *Macro* o *Modulo.*

Si può creare un modulo globale nell'editor di Visual Basic con il comando di menu *Inserisci*, *Modulo.*

Viene aggiunta nel riquadro ***Progetto*** una voce ***Moduli*** con il nome del modulo creato.
Per modificare un modulo esistente basta espandere la voce *Moduli* e selezionare il modulo desiderato.

Nota

Oltre ai moduli globali è possibile creare dei moduli di classe; in essi sono definite una o più classi (secondo il paradigma di programmazione orientato agli oggetti).

Conversione di macro in codice Visual Basic

Si può **convertire una macro** in una routine Visual Basic selezionando la macro nel riquadro di spostamento e usando lo strumento ***Converti macro in Visual Basic*** del gruppo *Macro* della barra *Strumenti database*. Viene creato un modulo che contiene la routine.
Una macro che gestisce un evento in una maschera può essere convertita in routine evento in visualizzazione struttura della maschera con lo strumento *Converti macro della maschera in Visual Basic* del gruppo *Macro* della barra *Strumenti database*; la routine evento viene inserita nel modulo della maschera.

Gli oggetti di Access

Alcuni **oggetti** di Access sono:

Application rappresenta l'applicazione Access;
Screen permette di accedere agli oggetti attivi sullo schermo;
Form ogni maschera aperta;
Report ogni report aperto;
Section ogni sezione di una maschera o report;
Control ogni controllo di una maschera o di un report;
DoCmd permette di eseguire le azioni da routine Visual Basic.

Alcuni oggetti dello stesso tipo sono raggruppati in **insiemi**, creati automaticamente, che sono a loro volta degli oggetti.

Forms insieme delle maschere aperte;
Reports insieme dei report aperti;
Controls insieme dei controlli di una maschera o un report;
AllForms insieme di tutte le maschere del database;
AllReports insieme di tutti i report del database;
AllQueries insieme di tutte le query del database;
AllTables insieme di tutte le tabelle del database;
Properties insieme di tutte le proprietà (come coppia nome-valore) di un oggetto *Form*, *Report* o *Control*.

Loescher Editore - Vietata la vendita e la diffusione

Proprietà degli **insiemi**:

Count numero di oggetti dell'insieme (vale 0 se l'insieme è vuoto);
Item(indice) indica l'oggetto dell'insieme che si trova alla posizione specificata da *Indice*; se *indice* è numerico è equivalente a *insieme(numero)*; se *indice* è il nome di un oggetto è equivalente a *insieme![nomeoggetto]*. .

Alcuni oggetti possiedono **proprietà** che sono un **riferimento** ad oggetti correlati:

Screen

ActiveForm	rappresenta la maschera attiva;
ActiveReport	rappresenta il report attivo;
ActiveDataSheet	rappresenta il foglio dati attivo;
ActiveControl	rappresenta il controllo attivo;
PreviousControl	rappresenta il controllo attivo prima di quello corrente.

Form e ***Report***

Module	rappresenta il modulo della maschera o del report;
Section	rappresenta una sezione della maschera o del report;
Controls	rappresenta l'insieme *Controls* della maschera o del report;
RecordsetClone	rappresenta una copia dei record di origine della maschera, cioè della tabella o della query su cui si basa la maschera (oggetto *Recordset* della maschera impostato tramite la proprietà *RecordSource*).

Control

Parent	rappresenta la maschera o il report in cui è presente il controllo;
Form	si utilizza per fare riferimento alla maschera contenuta in un controllo sottomaschera;
Report	si utilizza per fare riferimento al report contenuto in un controllo sottoreport.

Nota

Nei moduli di classe si può usare ***Me*** per fare riferimento all'oggetto in cui viene eseguito il codice.

Gli oggetti sono organizzati in una **gerarchia**: alcuni oggetti contengono altri oggetti. Tutti gli oggetti hanno la proprietà *Parent* che fa riferimento all'oggetto padre.

Per esempio ogni oggetto *Form* contiene un insieme *Controls* con tutti i controlli della maschera; la proprietà *Parent* di ogni controllo rappresenta la maschera.

Per identificare un oggetto ci si basa sulle relazioni esistenti tra gli elementi, usando gli operatori punto esclamativo (**!**) e punto (**.**).

L'**operatore punto esclamativo** (**!**) identifica un elemento di un insieme:

```
insieme!elemento
```

L'**operatore punto** (**.**) permette di accedere a proprietà e metodi di un oggetto (e quindi ai sottooggetti attraverso le proprietà relative):

```
oggetto.proprietà
```

Per una maschera

Per identificare una **maschera**, elemento dell'insieme *Forms*:

```
Forms![nomeMaschera]
```

Loescher Editore - Vietata la vendita e la diffusione

Per identificare un **controllo** di una maschera (un elemento dell'insieme *Controls* dei controlli della maschera):

```
Forms![Nomemaschera].Controls![Nome controllo]
```

Poiché l'insieme *Controls* è l'insieme predefinito per le maschere, non è necessario farvi riferimento in modo esplicito e si può scrivere:

```
Forms![nomeMaschera]![nomeControllo]
```

Per fare riferimento ad una **proprietà** di una maschera o di un controllo:

```
Forms![nomeMaschera].proprietà
Forms![nomeMaschera]![nomeControllo].proprietà
```

Nota

Si può fare riferimento a un controllo della maschera corrente semplicemente con *[nomeControllo]*, senza specificare il riferimento completo.

Riferimento alla casella di testo *NomeN* della maschera *Nazioni*:

```
Forms![Nazioni].Controls![NomeN]
```

oppure:

```
Forms![Nazioni]![NomeN]
```

Nota

Analogamente si possono definire gli identificatori relativi a report, controlli di report, proprietà di report e di controlli.

Sottomaschere

Per fare riferimento ad una sottomaschera bisogna utilizzare la proprietà ***Form*** del controllo della sottomaschera.

Riferimento alla **sottomaschera** corrente:

```
Forms![nomeMaschera]![controlloSottomaschera].Form
```

Identificatore di un **controllo** della sottomaschera:

```
Forms![nomeMaschera]![controlloSottomaschera].Form![nomeControllo]
```

oppure, in modo semplificato:

```
Forms![nomeMaschera]![controlloSottomaschera]![nomeControllo]
```

Nota

Il nome del controllo della sottomaschera è il valore della proprietà *Nome elemento* della scheda *Altro* della sottomaschera selezionata nella maschera principale (in visualizzazione struttura della maschera principale).

Nota

Analogamente per fare riferimento a un sottoreport si usa la proprietà *Report* del controllo del sottoreport.

Riferimento al controllo (casella di testo) *NomeN* della maschera *Nazioni* inserita nel controllo *SottomascheraNazioni* della maschera *ContinentiNazioni*:

Loescher Editore - Vietata la vendita e la diffusione

```
Forms![ContinentiNazioni]![SottomascheraNazioni].Form![NomeN]
```

o in modo semplificato come:

```
Forms![ContinentiNazioni]![SottomascheraNazioni]![NomeN]
```

Nel **codice** Visual Basic si può fare riferimento agli oggetti raggruppati in **insiemi** con le notazioni:

```
insieme("nomeOggetto")
```

dove il nome dell'oggetto può essere rappresentato anche da una variabile;

```
insieme(numero)
```

dove numero è un valore a partire da 0 che indica la posizione dell'oggetto nell'insieme;

```
insieme![nomeOggetto]
```

dove *nomeOggetto* può essere solo una costante e non va indicato tra virgolette.

Si può fare riferimento a maschere, report o controlli mediante **variabili** dichiarate come:

```
Dim nomeVariabile As tipoOggetto
```

dove *tipoOggetto* è uno dei tipi *Form*, *Report* o *Control*.

Alla variabile può essere assegnato un oggetto con l'istruzione ***Set***.

```
Dim mascheraAttiva as Form
Set mascheraAttiva = Screen.ActiveForm
```

Si può determinare il **tipo** specifico di controllo assegnato a una variabile di tipo *Control* usando come condizione nell'istruzione *If Then Else* una espressione nella forma:

```
TypeOf nomeOggetto Is tipoOggetto
```

```
If TypeOf nomeOggetto Is TextBox Then
   nomeOggetto.ForeColor = RBG(255,0,0)
End If
```

Per i controlli di una maschera è possibile utilizzare anche la proprietà *ControlType* che contiene una costante che indica il tipo di controllo.

8.8 LE APPLICAZIONI

Una **applicazione** Access è basata sugli eventi, ed è costituita da oggetti di database (tabelle, query, maschere, report, macro e moduli) uniti in un sistema coerente.

La realizzazione di un'applicazione consiste nella progettazione delle tabelle con le opportune relazioni e nella creazione delle maschere e dei report in base alle funzioni che l'applicazione deve svolgere, con il codice necessario (macro e routine).

Impostazioni di avvio

Si possono impostare le opzioni relative all'**avvio** di una applicazione nella sezione ***Database Corrente*** della finestra *Opzioni di Access* (che si apre facendo clic sul pulsante corrispondente dalla finestra del pulsante *Office*).

Loescher Editore - Vietata la vendita e la diffusione

È possibile specificare tra l'altro il **titolo** dell'applicazione, da visualizzare nella barra del titolo, e la **maschera** da aprire all'avvio.
Si possono scegliere anche impostazioni relative alla barra di stato e ai menu e fare in modo che non sia possibile accedere al **riquadro di spostamento** per la gestione degli oggetti.

Si può eseguire automaticamente una **macro** all'apertura del database creando la macro ***AutoExec***.

Quando si avvia l'applicazione, aprendo il database, vengono utilizzate le opzioni impostate nella sezione *Database corrente* della finestra *Opzioni di Access* e subito dopo vengono eseguite le azioni inserite nella macro *AutoExec*.

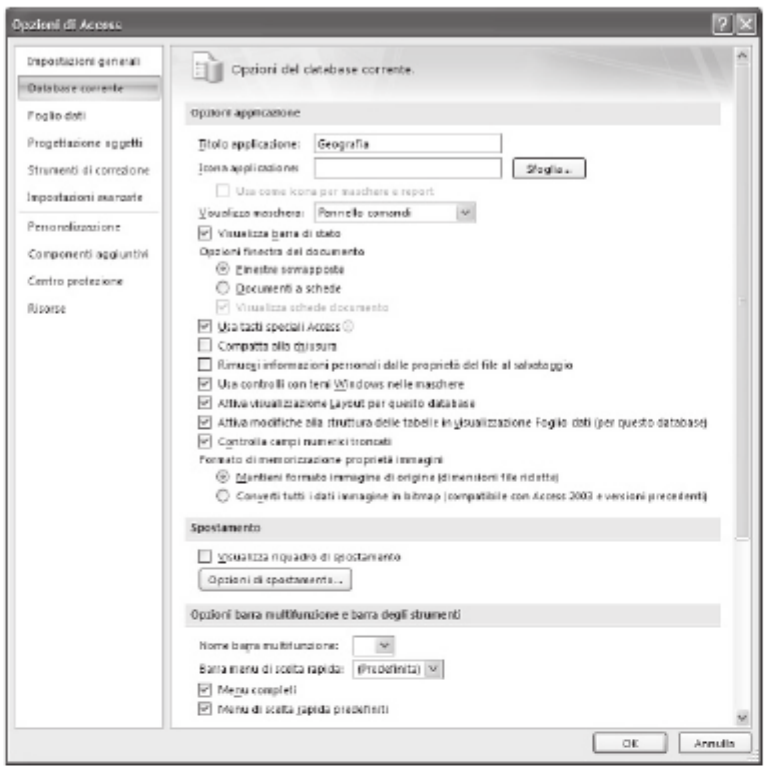


Figura 8.35 *Sezione Database corrente della finestra Opzioni di Access*

Nota

È possibile aprire il database senza utilizzare le opzioni di avvio né eseguire la macro *AutoExec* tenendo premuto il tasto *Shift* mentre si apre il database, ma solo se è attiva la casella di controllo *Usa tasti speciali Access* tra le opzioni.

Il pannello di comandi

Come prima maschera è possibile creare un **pannello dei comandi**.
Il pannello dei comandi è una maschera in cui è possibile inserire al massimo otto pulsanti che permettono di eseguire solo specifiche operazioni.

Si può creare un pannello dei comandi con lo strumento ***Gestione pannello comandi*** del gruppo *Strumenti database* della barra *Strumenti database*.

Viene creata una maschera *Pannello dei comandi* che può essere modificata in visualizzazione struttura. Le voci del pannello vengono registrate nella tabella ***Switchboard Items***; si può modificare il pannello dei comandi anche modificando la tabella.

Figura 8.36 *Scelta delle operazioni per i pulsanti del pannello dei comandi*

Figura 8.37 *Esempio di pannello dei comandi*

Loescher Editore - Vietata la vendita e la diffusione

Il formato .accde

Per fare in modo che gli utenti non possano modificare la struttura del database è possibile salvare e distribuire il database nel formato ***.accde*** (acronimo di ACCess Database Executable).

In un database in formato *.accde* non è possibile:

- aggiungere nuovi oggetti;
- modificare gli oggetti presenti perché non sono consentite le modalità struttura e layout;
- modificare o visualizzare le macro e il codice VBA perché tutto il codice è compilato.

Sono possibili invece tutte le operazioni sui dati previste dall'applicazione.

Per creare un file in formato *.accde* il contenuto deve essere attivo: nella *Barra dei messaggi* bisogna fare clic sul pulsante *Opzioni...* e nella finestra *Opzioni di protezione di Microsoft Office* selezionare l'opzione *Attiva il contenuto*. Questa operazione non è necessaria se il file è stato aperto da uno dei percorsi attendibili configurati nella scheda *Percorsi attendibili* della finestra *Centro di protezione*. **Nota**

Si può salvare il database in formato *.accde* con lo strumento ***Crea ACCDE*** del gruppo *Strumenti database* della barra *Strumenti database*; viene aperta la finestra *Salva con nome*.
Il file originale *.accdb* non viene modificato.
Non è possibile ritornare al formato *.accdb*. dal formato *.accde*.

È importante conservare il file originale *.accdb* per effettuare modifiche alla struttura del database, ricreando poi il file *.accde*. **Nota**

Nelle versioni Access 2002 e 2003 il formato non modificabile si chiama *.mde* (acronimo di Microsoft Database Executable). **Nota**

Protezione mediante password

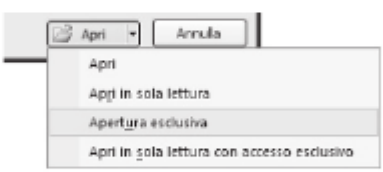


Figura 8.38 *Apertura esclusiva*

È possibile impostare una **password** per l'accesso al database.
Il database deve essere aperto in modalità esclusiva; nella finestra *Apri* (comando *Apri* del pulsante *Office*), dopo aver scelto il file, bisogna selezionare ***Apertura esclusiva*** dal menu che si ottiene facendo clic sulla freccia presente sul pulsante *Apri*.

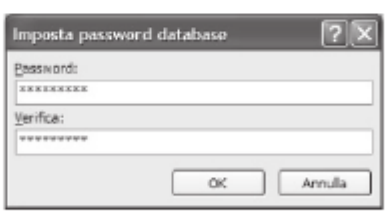


Figura 8.39 *Finestra Imposta password database*

La password si imposta o si modifica con lo strumento ***Crittografa mediante password*** del gruppo *Strumenti database* della barra *Strumenti database* che apre la finestra di dialogo *Imposta password database*.

Quando si imposta una password per l'accesso i dati vengono crittografati usando una chiave derivata dalla password. **Nota**

Condivisione di database

L'**accesso concorrente** da parte di più utenti viene realizzato separando il database in due file:

- il **back end**, contenente solo le tabelle;
- il **front end**, contenente tutti gli altri oggetti (query, maschere, report ecc.).

Loescher Editore - Vietata la vendita e la diffusione

Si può suddividere il database mediante una procedura guidata che si avvia con lo strumento ***Database di Access*** del gruppo *Sposta dati* della barra *Strumenti database*.
Per i nomi dei file back end per default viene aggiunto il suffisso ***_be*** al nome originale.
Il database di partenza diventa il front end; non contiene più le tabelle ma solo dei collegamenti alle tabelle del back end.

Il back end deve trovarsi su un server ed essere accessibile agli utenti.

Gli utenti accedono ai dati condivisi tramite una copia del front end. Il front end può essere personalizzato in base alle esigenze degli utenti.
Il front end può essere distribuito in formato *.accde* in modo che gli utenti non possano modificarlo.

Si può controllare la condivisione al database con l'opzione ***Modalità di apertura predefinita*** della sezione *Impostazioni avanzate* della finestra *Opzioni di Access* (che si apre facendo clic sul pulsante corrispondente dalla finestra del pulsante *Office*).

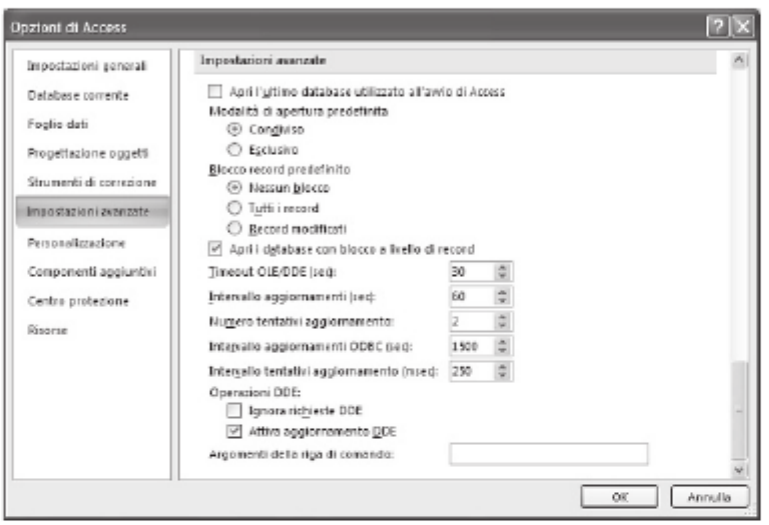


Figura 8.40
Sezione Impostazioni avanzate della finestra Opzioni di Access

L'opzione ***Blocco record predefinito*** permette di impostare il blocco dei record in caso di accesso concorrente; se il blocco è attivato un utente non può modificare un record già in uso fino a quando il record non viene rilasciato.

Le informazioni riguardanti l'accesso degli utenti al database sono registrate in un file con estensione ***.laccdb*** nella stessa cartella del database.

VERIFICA

QUESTIONARIO

1. Qual è l'estensione per i file di database di Access 2007?
2. Qual è l'estensione per i file di database delle versioni di Access fino alla 2003?
3. Quali sono gli oggetti di Access?
4. Che cos'è il riquadro di spostamento?
5. Che cos'è la finestra del database nelle versioni di Access fino alla 2003?
6. Quali tipi di visualizzazioni si possono scegliere per una tabella e che significato hanno?
7. Che cosa sono le relazioni?
8. Che cosa si intende per dynaset?
9. Che cos'è una query a campi incrociati?
10. Che cos'è una linea di join e a che cosa serve modificarne il tipo?

Loescher Editore - Vietata la vendita e la diffusione

11. Che cos'è la ricerca automatica?
12. Come si possono realizzare in Access le viste?
13. Che cosa sono le query di comando?
14. Che cosa sono le query di definizione e come possono essere realizzate?
15. Che cosa sono le maschere?
16. Quali tipi di visualizzazioni si possono scegliere per una maschera e che significato hanno?
17. Qual è la differenza tra una maschera con disposizione a colonne e una maschera con disposizione tabellare?
18. Come si può vedere su che tabella o query è basata una maschera?
19. Qual è la differenza tra controlli associati e controlli non associati?
20. Che cos'è una sottomaschera?
21. Che cosa sono i report?
22. Quali tipi di visualizzazioni si possono scegliere per un report e che significato hanno?
23. Quali sezioni si possono inserire in un report?
24. Che cos'è un sottoreport?
25. Che cosa sono le macro?
26. Che cosa sono le azioni e quali sono le azioni più comuni?
27. Che cos'è un gruppo macro?
28. Che cosa sono gli eventi e come si possono gestire con le macro?
29. Che cosa sono le routine evento?
30. Quali sono i vantaggi delle macro e quali quelli delle routine evento?
31. Come si possono eseguire routine dalle macro e come si possono eseguire azioni macro dalle routine?
32. Che cosa sono i moduli?
33. Qual è la differenza tra moduli di maschera o report e i moduli globali?
34. Qual è il significato degli operatori punto (.) e punto esclamativo (!) nella scrittura degli identificatori di oggetti?
35. Che cos'è una applicazione Access?
36. Come si possono modificare le impostazioni di avvio di una applicazione?
37. A che cosa serve la macro *AutoExec*?
38. Che cos'è un pannello dei comandi?
39. A che cosa serve il formato *.accde*?
40. Come si può impostare una password per l'accesso al database?
41. Come si può gestire l'accesso concorrente da parte di più utenti?
42. Come si possono impostare la modalità di apertura e il blocco record?

Esercizi

1 ..... Dato lo schema relazionale:
Studenti (Matricola, Cognome, Nome, Sesso, DataNascita, Indirizzo, Citta, Peso, Altezza, Occhiali)
dove *Matricola* è un campo contatore, *Sesso* può contenere i caratteri 'M' o 'F', *Occhiali* è un valore booleano;
Materie (IdMateria, DescrizioneM)
Voti (Studente, Materia, Data, Voto)
Voti.Studente è in integrità referenziale con *Studenti.Matricola*;
Voti.Materia è in integrità referenziale con *Materie.IdMateria*;

Loescher Editore - Vietata la vendita e la diffusione

1) creare le tabelle e inserire alcuni dati;
2) creare un indice sui campi *Cognome* e *Nome* della tabella *Studenti*;
3) creare le relazioni tra le tabelle;
4) creare una query sulla tabella *Studenti* per ricavare cognomi, nomi e indirizzi degli studenti che abitano in una data città;
5) creare una query sulla tabella *Studenti* per ricavare i dati degli studenti nati prima di una certa data e ordinarli in modo decrescente per data di nascita;
6) creare una query sulla tabella *Studenti* per ricavare i dati degli studenti che portano gli occhiali;
7) creare una query sulla tabella *Studenti* per ricavare i dati degli studenti con altezza e peso superiori a un valore parametrico (qualsiasi se si inserisce 0);
8) creare una query sulla tabella *Studenti* che ordini i dati in modo crescente rispetto all'altezza e decrescente rispetto al peso;
9) creare una query sulla tabella *Studenti* che ordini i dati in base al sesso e poi al cognome e nome in modo da ottenere i dati di maschi e femmine separatamente in ordine alfabetico;
10) creare una query sulla tabella *Studenti* per contare il numero di maschi e di femmine della classe;
11) creare una query sulla tabella *Studenti* per individuare l'altezza e il peso medi, minimi e massimi di maschi e femmine, separatamente;
12) creare una query a campi incrociati per calcolare il peso medio di maschi e femmine per ogni valore di statura;
13) creare una query a campi incrociati per determinare il numero di maschi e di femmine che provengono da ciascuna città;
14) creare una query sulle tabelle *Studenti*, *Voti* e *Materie* per ricavare cognome e nome dello studente, voto della prova, data e materia (decodificata);
15) creare una query che permetta di inserire i voti e di utilizzare la ricerca automatica per decodificare la materia e il nome dello studente;
16) creare una query per individuare i nomi degli studenti che non hanno mai preso insufficienze nelle prove;
17) creare una query per individuare per ogni studente le prove insufficienti;
18) creare una query per individuare i voti delle prove svolte in un certo intervallo di tempo per ciascun alunno;
19) creare una maschera per aggiornare i dati degli studenti con controlli opportuni in base al tipo di campi;
20) creare una maschera che riporti codice, cognome e nome di uno studente con una sottomaschera con l'elenco dei risultati delle prove dello studente, decodificando la materia;
21) creare un report per la stampa dei dati della tabella *Studenti*;
22) creare un report per la stampa dei dati della tabella *Studenti* divisi in maschi e femmine, che riporti il numero di maschi e il numero di femmine;
23) creare un report che stampi i voti riportati da ciascuna studente nelle varie prove;
24) creare un report che stampi l'elenco delle prove e il voto medio della classe in ogni prova;
25) creare una macro che apra la tabella *Studenti* e posizioni il cursore sulla casella relativa al peso;
26) nella maschera usata per aggiornare i dati degli studenti usare la proprietà di evento *Prima di aggiornare* per chiedere conferma prima di registrare le modifiche;
27) aggiungere alla maschera usata per aggiornare i dati degli studenti tre pulsanti di comando che permettano di ordinare i record in base a cognome e nome, alla data di nascita e all'altezza.

Loescher Editore - Vietata la vendita e la diffusione

2 ..... Per lo schema relazionale dell'esercizio 1 realizzare un'applicazione che permetta di gestire i dati degli studenti e dei voti nelle prove; deve essere possibile inserire e modificare i dati e ottenere tra l'altro:
- per ogni studente l'elenco dei voti delle prove nelle varie materie,
- per ogni studente l'elenco dei voti in una materia specifica;
- per ogni studente il voto medio in ogni materia;
- per una materia specifica l'elenco delle prove, il voto medio degli studenti in ciascuna prova e il numero delle sufficienze e delle insufficienze;
- per una prova specifica l'elenco dei voti riportati da tutti gli studenti o solo i voti insufficienti.

Inoltre:
- fare il backup del database;
- esportare i dati in formato XML;
- stampare la documentazione degli oggetti.

PROPOSTE DI LAVORO

1 ..... **Documenti multimediali**

Si vuole gestire un catalogo di testi, immagini, disegni e filmati che è possibile combinare insieme per realizzare documenti multimediali.
Ogni elemento deve essere corredato da termini descrittivi utili per la ricerca e informazioni sull'autore e dall'elenco dei documenti in cui è già stato utilizzato.
Deve essere possibile:
- stampare il catalogo con gli elementi suddivisi per tipo o per autore;
- dato un documento produrre la lista di tutti gli elementi dei vari tipi che lo compongono;
- scegliere gli elementi da assemblare in un nuovo documento, eseguendo ricerche per tipo, per autore o in base a uno o più termini di ricerca;
- individuare tutti i documenti che contengono uno o più elementi determinati.

2 ..... **Risultati olimpici**

Si vogliono memorizzare i dati delle medaglie olimpiche relative a tutte le discipline di tutte le olimpiadi svolte. Di ogni atleta o squadra che ha vinto una medaglia è conservata anche una fotografia e può essere disponibile un filmato della gara in cui la medaglia è stata vinta.
Deve essere possibile:
- ottenere il medagliere di ogni olimpiade (classifica delle nazioni partecipante ordinata in base al numero degli ori, per numero uguale in base agli argenti e per numero uguale in base ai bronzi);
- per una data olimpiade conoscere i vincitori di tutte le gare e a richiesta vedere la foto del vincitore o il filmato corrispondente;
- ottenere i record olimpici di tutte le discipline o solo di alcune specificate;
- ottenere i risultati in una disciplina ordinati per anno o dal migliore al peggiore;
- ottenere i risultati di una nazione in tutte le discipline o solo di alcune specificate in tutti gli anni o in un anno specificato;
- ottenere i risultati di un atleta nelle varie gare a cui ha partecipato.

3 ..... **Stato patrimoniale di un'azienda**

L'azienda possiede diversi conti (di cui sono noti saldo e la data di accensione) presso varie banche; possiede inoltre degli immobili, che possono essere terreni (estensione in m^2) o fabbricati (tipo, n° di vani, estensione in m^2), di ciascuno dei

Loescher Editore - Vietata la vendita e la diffusione

quali è noto il valore. L'azienda può aver acceso più ipoteche (di cui sono noti importo e scadenza), ciascuna concessa da una banca relativamente ad un immobile.
Deve essere possibile:
- ottenere la situazione patrimoniale dell'azienda;
- aprire o chiudere conti;
- acquistare o vendere fabbricati;
- accendere nuove ipoteche (la somma degli importi delle ipoteche relative ad un immobile non può superare il valore dell'immobile).

4 ..... Piccola rete di computer

Si vogliono gestire le informazioni sulla configurazione di una rete di computer.
Per ogni computer interessano: le caratteristiche hardware (tipo di processore, memoria, capacità del disco, periferiche collegate), il software installato (sistemi operativi e programmi per ciascun sistema operativo), l'indirizzo IP.
Si vuole registrare anche l'elenco degli utenti e dei gruppi utilizzati, le risorse condivise su ogni computer e l'elenco delle autorizzazioni concesse a ciascun utente.
Deve essere possibile:
- aggiungere o togliere un computer alla rete;
- modificare le caratteristiche hardware di un computer;
- modificare le installazioni software su ciascun computer;
- creare o eliminare utenti e gruppi e modificare l'assegnazione ai gruppi;
- modificare l'elenco delle risorse condivise su ciascun computer e le autorizzazioni concesse;
- modificare l'indirizzo IP di un computer controllando che il nuovo indirizzo non sia già assegnato;
- individuare i computer dotati di certe caratteristiche hardware o software;
- individuare tutte le autorizzazioni concesse a un certo utente;
- individuare tutti gli utenti che possono accedere a una certa risorsa.

5 ..... Azienda di produzione

Una azienda produce prodotti finiti a partire da materie prime e da prodotti di lavorazioni intermedie (semilavorati); ogni semilavorato può utilizzare a sua volta materie prime e altri semilavorati e così via.
Per ogni prodotto (finito o semilavorato) devono essere memorizzate le quantità di materie prime e semilavorati necessari, nell'unità di misura opportuna, e il tempo necessario alla realizzazione.
Per ogni materia prima e semilavorato sono memorizzate le quantità presenti in magazzino e il tempo necessario per l'approvvigionamento.
Deve essere possibile:
- individuare l'elenco di tutte le materie prime necessarie a realizzare un prodotto, con relativa quantità, tenendo conto anche di tutte quelle necessarie a realizzare i semilavorati intermedi;
- individuare il tempo necessario a realizzare un prodotto considerando i semilavorati e le materie prime disponibili e gli eventuali tempi di approvvigionamento;
- individuare l'elenco dei prodotti che usano una certa materia prima in qualunque fase della produzione.

6 ..... Albero genealogico

Si vogliono memorizzare le informazioni relative all'albero genealogico di una famiglia con relative fotografie, se disponibili.
Deve essere possibile navigare nell'albero visualizzando dati e foto di ciascuno.
Deve essere possibile anche data una persona ottenere a richiesta l'elenco di genitori, fratelli, nonni, zii o cugini di primo grado.

Loescher Editore - Vietata la vendita e la diffusione

Modulo 4

Accesso a database

9 Accesso ai database dalle applicazioni

10 Accesso a database via Web

Loescher Editore - Vietata la vendita e la diffusione

9 Accesso ai database dalle applicazioni

Prerequisiti

- Linguaggi Visual Basic e Java
- Database relazionali e linguaggio SQL
- Modalità di accesso ai database

Obiettivi

Conoscenze

- Tecnologie di accesso ai database (ODBC, OLE DB, ADO, ADO.NET, JDBC)
- Principali classi di ADO.NET, in particolare: DBCommand, DataReader, DataSet, DataAdapter
- Data binding con i controlli comuni e con il controllo DataGridView
- Principali classi e interfacce del package java.sql in particolare: DriverManager, Connection, Statement, ResultSet, PreparedStatement, DatabaseMetaData, ResultSetMetaData

Abilità/capacità

- Creare una origine dati ODBC
- Realizzare applicazioni Visual Basic che interagiscono con il database in modalità connessa e in modalità disconnessa, utilizzando anche tecniche di data binding
- Realizzare applicazioni Java che interagiscono con il database, utilizzando anche metadati

Loescher Editore - Vietata la vendita e la diffusione

9.1 TECNOLOGIE DI ACCESSO AI DATABASE

Esistono molte soluzioni per l'accesso ai database. Lo scopo principale è di permettere ad una applicazione di accedere a database diversi allo stesso modo, senza che siano necessarie modifiche all'applicazione.
Le principali tecnologie per *Windows* sono ODBC, OLE DB, ADO e ADO.NET.
Per quanto lo sviluppo di applicazioni in Java è disponibile la tecnologia JDBC.

Tecnologie di accesso ai database

- **ODBC** (Open DataBase Connectivity) interfaccia standard per l'accesso a database mediante SQL: lo stesso programma può accedere a qualsiasi DBMS dotato di un driver ODBC;
- **OLE DB** (Object Linking and Embedding for DataBase) interfaccia generale che permette di accedere a sorgenti dati generiche (non solo DBMS);
- **ADO** (ActiveX Data Object) permette di sfruttare i servizi OLE DB utilizzando un'interfaccia record oriented;
- **ADO.NET** adatta ADO alla piattaforma .NET; offre un'interfaccia set oriented e introduce i data adapter;
- **JDBC** soluzione Java.

Driver per database ODBC (Open DataBase Connectivity)

Le applicazioni possono accedere a un database tramite un **driver ODBC** che permette di interagire con il database utilizzando istruzioni SQL.

I **driver per database** sono programmi che permettono di accedere a un database in modo standard, senza dover scrivere istruzioni specifiche per ogni diverso tipo di database.

Sul computer su cui si utilizza l'applicazione è necessaria l'installazione del driver ODBC e la configurazione di una origine dati ODBC.

Una **origine dati ODBC**, o **DSN** (Data Source Name), identifica il database indicando dove si trova il database e quale driver ODBC si vuole utilizzare per accedervi.

L'**applicazione** specifica l'origine dati ODBC da utilizzare e ogni volta che si esegue un'operazione, il driver ODBC la traduce in un formato compatibile con il database.

Creazione di una origine dati ODBC

Si può **creare una origine dati ODBC** dal *Pannello di controllo*, *Strumenti di amministrazione*, *Origine dati (ODBC)*. Si apre la finestra *Amministrazione origine dati ODBC*.
La scheda *Driver* permette di vedere i driver ODBC installati sul sistema; si può utilizzare il driver *Microsoft Access Driver (*.mdb)* per database creati con *Microsoft Access*.
Bisogna creare un DSN utente, di sistema o su file selezionando la scheda relativa.
Un **DSN utente** è disponibile solo a uno specifico utente, un **DSN di sistema** a tutti gli utenti del sistema, un **DSN su file** a chiunque abbia installato gli stessi driver (anche utenti remoti).
Dopo aver selezionato la scheda si fa clic sul pulsante *Aggiungi* e si seleziona il driver desiderato. Selezionando il driver per *Microsoft Access* si apre la finestra *Configurazione ODBC per Microsoft Access* in cui si deve specificare il nome del DSN e selezionare il database da utilizzare con il pulsante *Seleziona...* o *Crea...*; il database si può trovare anche su un altro computer collegato in rete.

Loescher Editore - Vietata la vendita e la diffusione

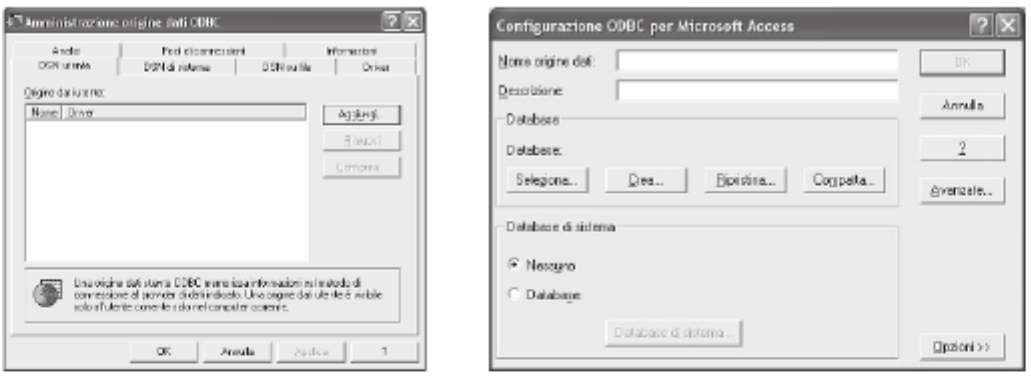


Figura 9.1 *Le finestre Amministrazione origine dati ODBC e Configurazione ODBC per Microsoft Access*

OLE DB

OLE DB (Object Linking and Embedding for DataBase) è un'interfaccia generale, molto più flessibile di ODBC, che offre una modalità uniforme di accesso a dati di tipo diverso.

> **OLE DB** supporta tutte le funzionalità SQL definite in ODBC ma, oltre che ai database, permette di accedere anche a molti altri tipi di dati (come file system o posta elettronica).

OLE DB è un'**API C++** basata sulla tecnologia **COM**.

ADO

> **ADO** (ActiveX Data Object) è un'**interfaccia di alto livello** per accedere ai servizi di OLE DB, che può essere usata nei diversi linguaggi di programmazione disponibili per lo sviluppo di applicazioni *Windows*.

ADO fornisce tre classi di oggetti principali:

Connection	rappresenta una connessione al database;
Command	rappresenta una istruzione SQL che si vuole fare eseguire;
RecordSet	è un insieme di tuple (set di record) restituito da un comando SQL.

Per gestire dei dati bisogna creare un oggetto ***RecordSet***.
Per prima cosa bisogna specificare una connessione a un database, poi bisogna specificare i dati su cui si intende lavorare indicando una tabella, una query o un'istruzione SQL che dica come prelevare i dati e infine bisogna creare il relativo *RecordSet*.

Ognuno dei tre oggetti di ADO permette stabilire una connessione e di creare un oggetto *RecordSet*.

> Si può fare in modi diversi partendo da ciascuno dei tre oggetti. **Nota**

Per lavorare sul *RecordSet* si usa un cursore.
Il **cursore** è un puntatore della riga corrente all'interno del set di record creato.
I cursori possono essere statici o dinamici; i cursori dinamici, a differenza dei cursori statici, sono in grado di rilevare aggiunte e cancellazioni operate da altri utenti.
Sui *RecordSet* creati con gli oggetti *Connection* e *Command* sono disponibili solo cursori statici di tipo forward only (che permettono solo di scorrere i record in avanti).
Per operazioni più complesse bisogna creare un *RecordSet* con il metodo *Open()* dell'oggetto *RecordSet*.

Loescher Editore - Vietata la vendita e la diffusione

DAO (Data Access Object)

ADO è un'evoluzione del precedente modello **DAO** (Data Access Object).
In DAO gli oggetti sono organizzati rigidamente in una **struttura gerarchica**.
L'oggetto *DBEngine* rappresenta il modulo di gestione di database Microsoft Jet e contiene e controlla tutti gli altri oggetti.
Quando viene avviata l'applicazione, nell'oggetto *DBEngine* viene creato un oggetto *Workspace* che rappresenta l'area di lavoro, e all'interno di questo un oggetto *Database* che rappresenta il database aperto.
Ciascun oggetto *Database* contiene oggetti di tipo *Tabledef* (tabelle salvate nel database), *Querydef* (query salvate nel database) e *Recordset* (insieme dei record contenuti in una tabella o individuati da una query).
Gli oggetti *Recordset* permettono di eseguire operazioni di gestione dei dati.

ADO.NET

In **ADO.NET** la connessione al database avviene mediante **provider** che permettono di accedere ai dati del database; esistono provider diversi in base al tipo di database:

- il provider SQL è specifico per SQL Server a partire dalla versione 7.0;
- il provider OLE DB funziona con tutti gli altri tipi di dati.

L'analogo dell'oggetto *RecordSet* di ADO in ADO.NET è ***DataReader*** che permette di effettuare letture in **modalità connessa** al database con un cursore forward only; si può ottenere un oggetto *DataReader* eseguendo un comando SQL tramite il metodo *ExecuteReader()* di un oggetto ***DbCommand***.

La novità di ADO.NET è costituita dall'oggetto ***DataSet*** che rappresenta un database in memoria: un contenitore di dati su cui si può lavorare in **modalità disconnessa**.

Un *DataSet* contiene oggetti di vario tipo tra cui *DataTable* che rappresentano le tabelle; non sono richiesti join per usare dati suddivisi in più tabelle; l'accesso agli elementi dei *DataSet* avviene con i normali meccanismi di accesso a collezioni di oggetti.
Il coordinamento tra i *DataSet* e le sorgenti dati avviene tramite i ***DataAdapter***; i metodi principali di un *DataAdapter* sono *Fill()* per caricare i dati dalla sorgente nel *DataSet* e *Update()* per aggiornare la sorgente con le variazioni apportate al *DataSet*.
I *DataSet* inoltre permettono di convertire facilmente i dati da e verso il formato XML.

9.2 ACCESSO A DATABASE DA VISUAL BASIC CON ADO.NET

Connessione al database e modalità di lavoro

ADO.NET usa classi diverse per interagire con uno specifico database, raggruppate in namespace specifici:

System.Data.SqlClient	per Microsoft SQL Server versione 7.0 o superiori;
System.Data.OleDb	per DBMS OLE DB come Microsoft Access;
System.Data.Odbc	per DBMS basati su ODBC.

Per aprire una connessione a un database bisogna importare il namespace, creare una istanza della classe per la connessione e aprire la connessione.

Loescher Editore - Vietata la vendita e la diffusione

Per un database Access, dopo aver importato *System.Data.OleDb*:

```
Dim origine As OleDbConnection
origine = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source= ...")
origine.Open()
```

Poi si può lavorare in **due modalità**:

- rimanendo **connessi** al database usando un oggetto ***DbCommand*** per eseguire l'operazione (esegue un comando SQL e restituisce un cursore);
- usando un ***DataAdapter*** che carica i dati in un contenitore come ***DataSet*** su cui poi si può lavorare in **modalità disconnessa**.

DbCommand

DbCommand (in Access ***OleDbCommand***) permette di specificare il **comando SQL** da eseguire.

Per **eseguire** il comando si deve usare uno dei metodi:

ExecuteReader()	per effettuare una lettura con il comando SELECT; restituisce un cursore *DataReader* per leggere i dati;
ExecuteNonQuery()	per eseguire un comando di aggiornamento; restituisce il numero di righe su cui il comando ha avuto effetto.

Un ***DataReader*** permette di effettuare letture in modalità connessa al database con un cursore forward only.

Proprietà

HasRows	indica se ci sono dati;
Item	ricava il valore di una colonna nel formato originale.

Metodi

Read()	sposta il puntatore sul record successivo; restituisce *false* se non ci sono altri dati.

Esempio 9.1

Lettura con *DataReader*.

Figura 9.2 *Esecuzione dell'esempio*

```
Imports System.Data.OleDb
Public Class frmLettura

Private Sub cmdEsegui_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdEsegui.Click
    Dim s As String = ""
    Dim origine As OleDbConnection
    Dim selezione As OleDbCommand
    Dim dati As OleDbDataReader

    origine = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\esempiVB\cap.mdb")
    origine.Open()
    selezione = New OleDbCommand("SELECT * FROM elenco", origine)
    dati = selezione.ExecuteReader()
    While dati.Read()
        s &= dati.Item("citta") & " " & dati.Item("cap") & Chr(13) & Chr(10)
    End While
```

Loescher Editore - Vietata la vendita e la diffusione

```
        lblMostra.Text = s
        origine.Close()
End Sub

...
End Class
```

Esempio 9.2

Inserimento di dati in modalità connessa.

```
Imports System.Data.OleDb
Public Class frmInserimento
...
Private Sub cmdInserisci_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdInserisci.Click
        Dim origine As OleDbConnection
        Dim comando As OleDbCommand
        origine = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\esempiVB\cap.mdb")
        origine.Open()
        comando = New OleDbCommand("INSERT INTO elenco (citta, cap) VALUES ('" & txtCitta.Text & "', '"
& txtCap.Text & "')", origine)
        Try
                comando.ExecuteNonQuery()
        Catch ee As OleDbException
                MsgBox(ee.Message, MsgBoxStyle.Critical, "Errore")
        End Try
        origine.Close()
End Sub

...
End Class
```

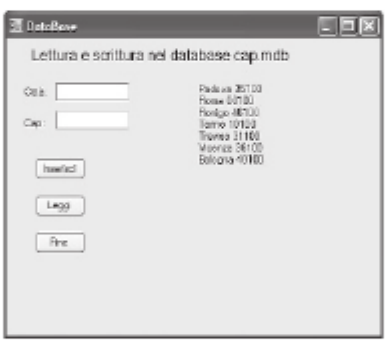


Figura 9.3 *Esecuzione dell'esempio*

DataSet e DataAdapter

Un ***DataSet*** rappresenta un database in memoria; è un contenitore di dati disconnesso.
È possibile creare direttamente le tabelle del database o caricare i dati da un database esistente.
Dopo che il *DataSet* è stato impostato con i dati, si può lavorare in **modalità disconnessa**.

Un ***DataAdapter*** si comporta come un ponte bidirezionale tra una sorgente dati e l'oggetto *DataSet*.
Per caricare i dati da un database bisogna aprire una connessione col database e creare un oggetto *DataAdapter*.
La classe *DataAdapter* presenta diverse specializzazioni secondo le caratteristiche della sorgente dati:

SqlDataAdapter	per Microsoft SQL Server versione 7.0 o superiori;
OleDbDataAdapter	per DBMS OLE DB come Microsoft Access;
OdbcDataAdapter	per DBMS basati su ODBC.

Poi bisogna specificare i dati da raccogliere dal database e inserirli nel *DataSet* col metodo ***Fill()***.

Loescher Editore - Vietata la vendita e la diffusione

Per un database Access, dopo aver importato *System.Data.OleDb*:

```
Dim selezione As OleDbDataAdapter
selezione = New OleDbDataAdapter("SELECT * FROM elenco", origine)
selezione.Fill(dati, "elenco")
```

Poiché il *DataSet* lavora in modalità disconnessa, se si fanno delle modifiche e si vogliono rendere effettive nel database bisogna aggiornare il database fisico.
Il metodo ***Update()*** del DataAdapter aggiorna il database fisico in base al contenuto della tabella in memoria specificata; funziona invocando comandi INSERT, UPDATE e DELETE per ciascuna riga della tabella inserita, modificata o cancellata.

Esempio 9.3

Lettura con *DataSet*.

```
Imports System.Data.OleDb
Public Class frmDataSet

Private Sub cmdEsegui_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
cmdEsegui.Click
    Dim s As String = ""
    Dim origine As OleDbConnection
    Dim selezione As OleDbDataAdapter
    Dim dati As DataSet
    Dim lettore As DataTableReader
    origine = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\esempiVB\cap.mdb")
    origine.Open()
    selezione = New OleDbDataAdapter("SELECT * FROM elenco", origine)
    dati = New DataSet()
    selezione.Fill(dati, "elenco")
    origine.Close()
    lettore = dati.CreateDataReader()
    While lettore.Read()
        s &= lettore(0) & " " & lettore(1) & Chr(13) & Chr(10)
    End While
    lblMostra.Text = s
    lettore.Close()
End Sub

...
End Class
```

Figura 9.4
Esecuzione dell'esempio

Data binding

Il **data binding** (associazione dati) permette di associare dati a controlli o a proprietà di controlli. Si può stabilire il modo in cui i dati vengono spostati da una sorgente alla pagina o viceversa e si possono utilizzare i dati nelle elaborazioni, ma anche modificarli e aggiornare la sorgente; come sorgente si possono usare database, ma anche *DataSet*, *DataReader* e oggetti personalizzati come *Collection*, *Dictionary* e *Array*).

Connessione a una origine dati

Per creare una connessione a una origine dati in un progetto basta scegliere dal menu ***Dati*** il comando *Aggiungi nuova origine dati...* che apre la finestra *Configurazione guidata origine dati*.

Loescher Editore - Vietata la vendita e la diffusione

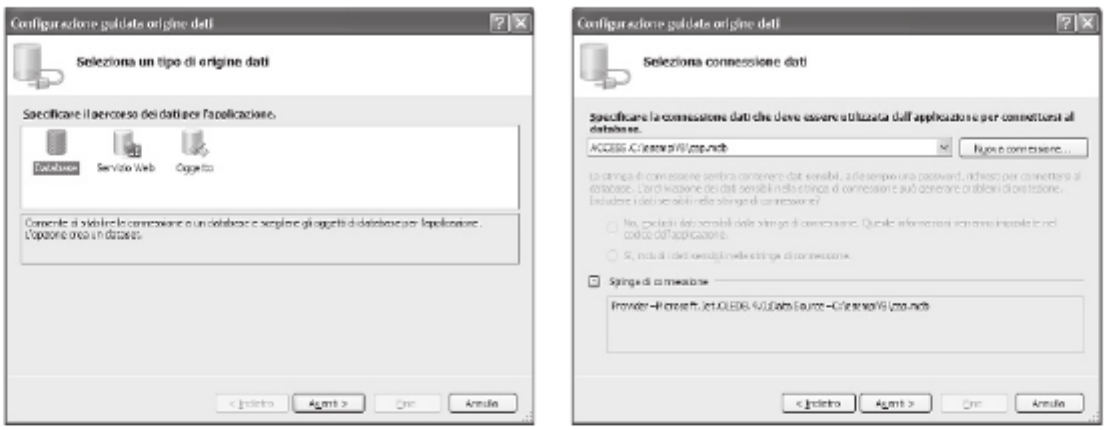


Figura 9.5 *Primi passi della Configurazione guidata origine dati*

Figura 9.6
La finestra Aggiungi connessione

Per default è già selezionato *Database*; questa scelta consente di stabilire la connessione a un database e di scegliere le tabelle per creare un *DataSet* da usare nell'applicazione.

Facendo clic su *Avanti* e nella finestra successiva sul pulsante *Nuova connessione...* si apre la finestra *Aggiungi connessione* in cui è possibile scegliere l'origine dati (pulsante *Modifica...*) cioè il provider per la connessione (*File di database Microsoft Access*, oppure *File di database Microsoft SQL Server*) e il file di database (pulsante *Sfoglia...*); il pulsante *Test connessione* permette di verificare che la connessione funzioni.
Facendo clic sul pulsante *OK* si torna alla finestra precedente in cui, nel riquadro *Stringa di connessione* è possibile vedere il provider creato.
Facendo clic su *Avanti* viene chiesto di scegliere se si desidera copiare il file di dati nel progetto corrente; normalmente è bene rispondere *No* (scegliendo *Sì* gli aggiornamenti sono temporanei all'esecuzione dell'applicazione e non vengono riportati nel database originale).

La finestra successiva permette di specificare un nome per la stringa di connessione e di salvarla nel file di configurazione dell'applicazione.

L'ultima finestra permette di specificare gli oggetti del database da inserire nel *DataSet*. Si può selezionare una tabella o i singoli campi. Nella casella in basso si può specificare il nome del *DataSet*.

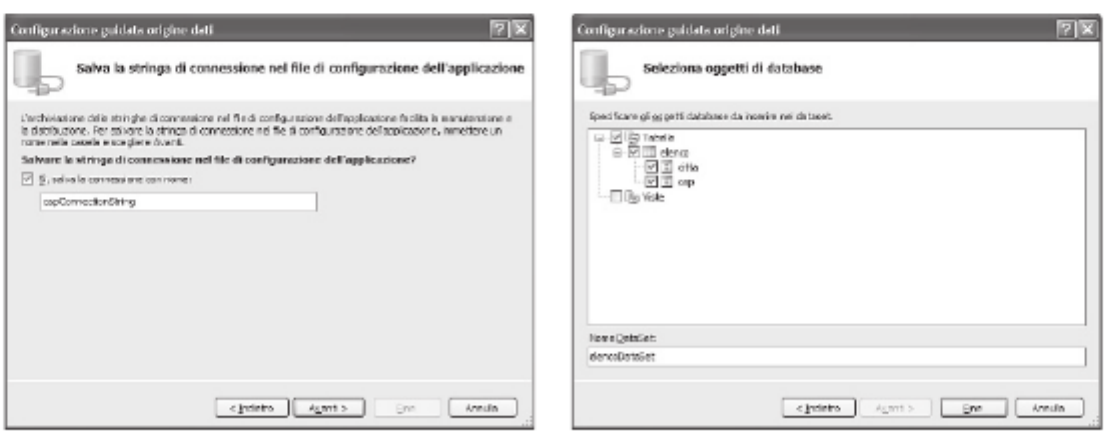


Figura 9.7 *Ultimi passi della Configurazione guidata origine dati*

Loescher Editore - Vietata la vendita e la diffusione

Nel progetto viene creata automaticamente la routine evento *Load* del form, che richiama il metodo *Fill* per caricare il *DataSet*.

```
Private Sub frmBN_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    'TODO: questa riga di codice carica i dati nella tabella 'CapDataSet.elenco'. È possibile spostarla o
    'rimuoverla se necessario.
    Me.ElencoTableAdapter.Fill(Me.CapDataSet.elenco)
End Sub
```

Scegliendo da menu *Dati*, *Mostra origine Dati* si apre la sottofinestra ***Origini dati*** che mostra i *DataSet* relativi alle origini dati create.

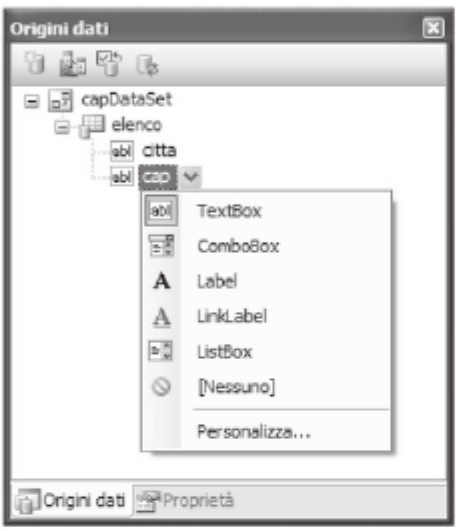


Figura 9.8 *La finestra Origini dati*

Data binding con controlli comuni

Nella finestra *Origini dati*, espandendo una tabella si vede l'elenco dei campi; a fianco di ogni campo una freccia permette di espandere un elenco da cui scegliere il tipo di controllo che si vuole utilizzare collegato a quel campo (tra cui *TextBox*, *ComboBox*, *Label* e *ListBox*).

Trascinando un campo nel form, viene creato il controllo selezionato con a fianco l'etichetta col nome del campo. Alla creazione del primo campo viene creato anche un oggetto ***BindingNavigator***: una barra di navigazione con pulsanti per spostarsi avanti, indietro, all'inizio e alla fine, una casella per specificare il numero di record, il numero di record presenti e pulsanti per inserire un nuovo record, eliminare un record e salvare le modifiche apportate.

L'inserimento e l'eliminazione agiscono sul *DataSet*; per il salvataggio viene creata automaticamente una routine evento che usa gli oggetti *BindingSource* e *TableAdapter*.

```
Private Sub ElencoBindingNavigatorSaveItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ElencoBindingNavigatorSaveItem.Click
    Me.Validate()
    Me.ElencoBindingSource.EndEdit()
    Me.ElencoTableAdapter.Update(Me.CapDataSet.elenco)
End Sub
```

Esempio 9.4

Data binding con caselle di testo.

Nota Tutto il codice è generato in modo automatico.

```
Public Class frmBN

Private Sub ElencoBindingNavigatorSaveItem_Click(ByVal sender_
As System.Object, ByVal e As System.EventArgs) Handles
ElencoBindingNavigatorSaveItem.Click
    Me.Validate()
    Me.ElencoBindingSource.EndEdit()
    Me.ElencoTableAdapter.Update(Me.CapDataSet.elenco)
End Sub
```

Figura 9.9 *Esecuzione dell'esempio*

Loescher Editore - Vietata la vendita e la diffusione

```
Private Sub frmBN_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    'TODO: questa riga di codice carica i dati nella tabella 'CapDataSet.elenco'. È possibile spostarla o
rimuoverla se necessario.
    Me.ElencoTableAdapter.Fill(Me.CapDataSet.elenco)
End Sub

End Class
```

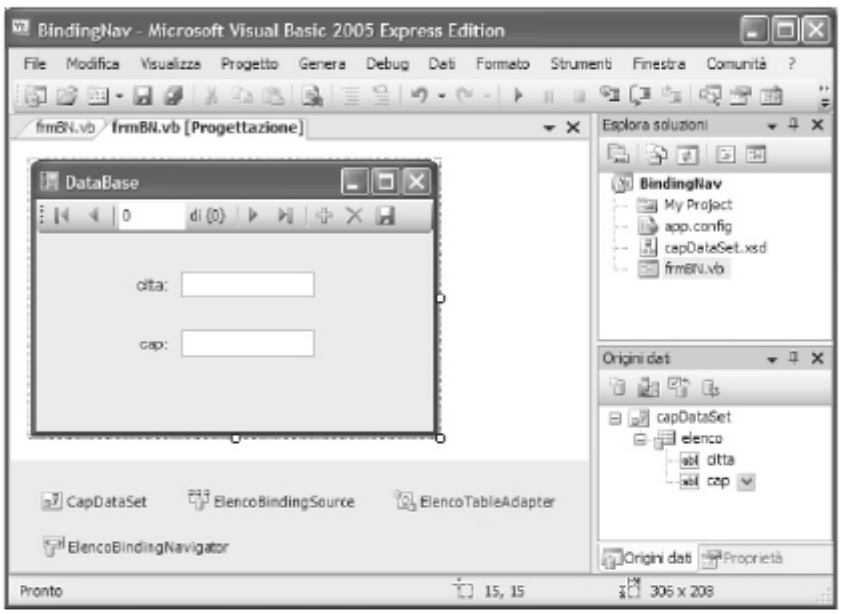


Figura 9.10 *La finestra di progetto, gli oggetti elencati in fondo sono stati creati automaticamente*

Data binding con DataGridView

È possibile associare a un *DataSet* un controllo *DataGridView*; basta trascinare sul form un controllo *DataGridView*, aprire la finestra *Attività di DataGridView* e scegliere l'origine dati già creata; automaticamente viene creato il controllo con le colonne della tabella e le relative intestazioni (i nomi dei campi).

Figura 9.11 *Scelta dell'origine dati per un controllo DataGridView*

La proprietà ***DataSource*** del *DataGridView* viene impostata automaticamente al *BindingSource* creato in automatico alla creazione del *DataSet* (anche il *TableAdapter* viene creato automaticamente).

In esecuzione il controllo visualizza i dati e fornisce una barra di scorrimento. Si possono modificare o aggiungere dati ma solo sul *DataSet* (non sul database originale).

Per fare in modo che le modifiche apportate vengano salvate nel database bisogna aggiungere controlli e codice opportuno (per esempio un pulsante *Aggiorna* e la routine evento *cmdAggiorna*).

Loescher Editore - Vietata la vendita e la diffusione

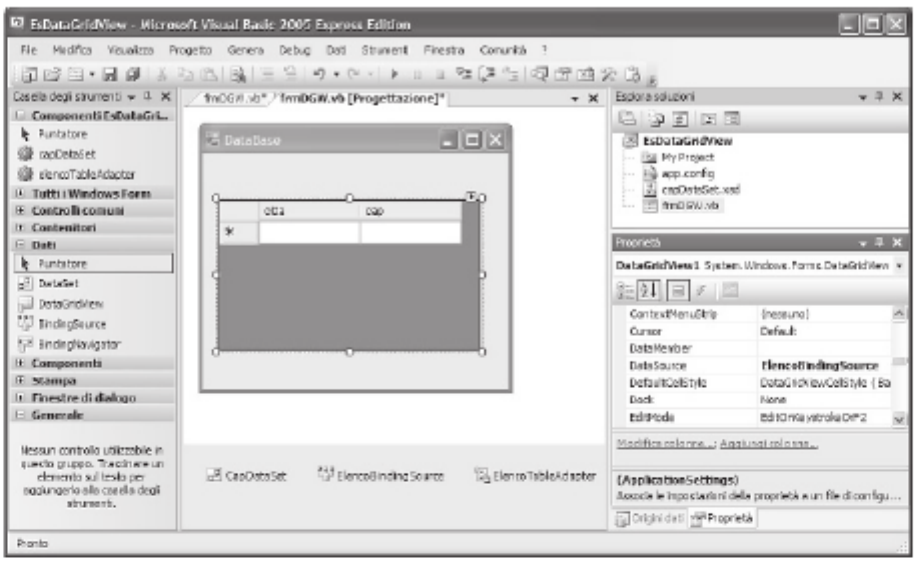


Figura 9.12 *La finestra di progetto per il data binding con un oggetto DataGridView*

Esempio 9.5

Data binding con *DataGridView*.

```
Public Class frmDGV
...

Private Sub frmDGV_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
    'TODO: questa riga di codice carica i dati nella tabella
    'oggElencoDS.elenco. È possibile spostarla o rimuoverla
    'se necessario.
    Me.ElencoTableAdapter.Fill(Me.oggElencoDS.elenco)
End Sub

Private Sub cmdAggiorna_Click(ByVal sender As System.Object,_
ByVal e As System.EventArgs) Handles cmdAggiorna.Click
    Me.Validate()
    Me.ElencoBindingSource.EndEdit()
    Me.ElencoTableAdapter.Update(Me.oggElencoDS.elenco)
End Sub

End Class
```

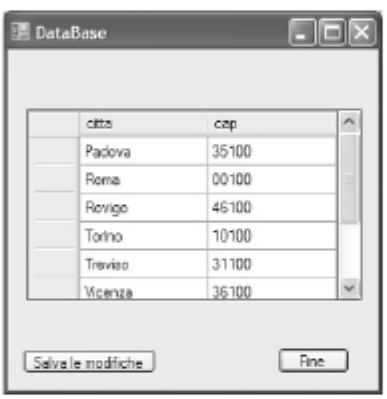


Figura 9.13 *Esecuzione dell'esempio*

9.3 ACCESSO A DATABASE DA JAVA: DRIVER JDBC

Servlet, pagine JSP e applicazioni Java possono accedere a un database tramite un **driver JDBC** che permette di interagire con il database utilizzando istruzioni SQL.

Ci sono vari tipi di driver JDBC, alcuni dei quali non richiedono nessuna installazione sul computer su cui si utilizza l'applicazione.
Uno dei driver JDBC più utilizzati sui sistemi *Windows* è il **driver JDBC-ODBC** che permette di accedere ai database dall'API Java attraverso i driver ODBC (Open DataBase Connectivity), ma che richiede, sul computer su cui si utilizza l'applicazione, l'installazione del driver ODBC e la configurazione di una origine dati ODBC.

Loescher Editore - Vietata la vendita e la diffusione

9.4 CREAZIONE DI APPLICAZIONI CON JDBC

L'**API JDBC** è l'insieme delle classi Java necessarie per accedere ai database; le classi dell'API JDBC si trovano nel package *java.sql.*

Caricamento del driver e creazione della connessione

In una applicazione per prima cosa si deve **caricare il driver** da utilizzare con il metodo statico *forName()* della classe *Class*.

```
static Class forName(String className)
```

Nota Il metodo *forName()* forza il caricamento di una classe.

Il driver JDBC-ODBC è individuato dalla stringa *sun.jdbc.odbc.JdbcOdbcDriver*.

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

La classe ***DriverManager***, che permette di gestire i driver, offre il metodo statico ***getConnection()*** che **apre una connessione** con il database specificato, utilizzando il driver caricato.

```
static Connection getConnection(String url) throws SQLException
```

Il parametro *url* da passare al metodo *getConnection()* deve avere la forma:

```
jdbc:subprotocol:subname
```

dove *subprotocol* indica il driver da utilizzare e *subname* è il nome dell'origine dati.

Il metodo restituisce un oggetto di tipo ***Connection*** che rappresenta la **connessione**, cioè una **sessione di lavoro** sul database.

```
Connection connessione = DriverManager.getConnection("jdbc:odbc:origine");
```

Operazioni sul database

Per **eseguire delle operazioni** sul database si deve creare un oggetto di tipo ***Statement***.

Si può creare un oggetto di tipo *Statement* usando il metodo ***createStatement()*** dell'interfaccia *Connection*:

```
Statement st = connessione.createStatement();
```

Statement è un'interfaccia che offre i metodi per eseguire operazioni sul database:

executeQuery()	per operazioni di lettura;
executeUpdate()	per operazioni di modifica.

Le operazioni sul database sono effettuate tramite **comandi SQL**.

Interrogazioni

Per eseguire **interrogazioni** si usa il metodo ***executeQuery()*** dell'interfaccia ***Statement***.
Il metodo *executeQuery()* ha come parametro una stringa con una istruzione SQL *SELECT* e restituisce un oggetto di tipo *ResultSet*.

```
ResultSet rs = st.executeQuery("SELECT ...");
```

Un oggetto di tipo ***ResultSet*** fornisce l'accesso a una tabella composta da righe (record) e colonne (campi) attraverso un cursore.

Loescher Editore - Vietata la vendita e la diffusione

Il **cursore** indica un record della tabella; quando viene creato l'oggetto *ResultSet*, il cursore punta al record precedente al primo; si può spostare il cursore sul record successivo o precedente o su un record qualsiasi specificandone l'indice; il metodo ***next()*** sposta il cursore sul record successivo e restituisce *true* se è stato selezionato un record valido, *false* se non ci sono altri record.

Quando il cursore si trova su un record si possono **leggere i campi** del record; per leggere i campi del record si usano metodi specifici in base al tipo di valore contenuto nel campo; questi metodi richiedono come parametro il nome del campo o il numero della colonna:

double **getDouble**(String columnName)	legge un valore *double* dato il nome del campo;
double **getDouble**(int columnIndex)	legge un valore *double* dato il numero della colonna;
int **getInt**(String columnName)	legge un valore *int* dato il nome del campo;
int **getInt**(int columnIndex)	legge un valore *int* dato il numero della colonna;
Object **getObject**(String columnName)	legge un oggetto dato il nome del campo;
Object **getObject**(int columnIndex)	legge un oggetto dato il numero della colonna;
String **getString**(String columnName)	legge una stringa dato il nome del campo;
String **getString**(int columnIndex)	legge una stringa dato il numero della colonna;
...	

Si possono stampare tutti i dati di una colonna con un ciclo del tipo:

```
while(rs.next()) {
    String campo = rs.getString("NomeColonna");
    System.out.println("NomeColonna : " + campo);
}
```

Aggiornamenti

Per eseguire operazioni di **inserimento**, **modifica** o **cancellazione** di dati si usa il metodo ***executeUpdate()*** dell'interfaccia ***Statement***.
Il metodo *executeUpdate()* ha come parametro una stringa con una istruzione SQL *INSERT*, *UPDATE* o *DELETE*.

```
st.executeUpdate("INSERT INTO ...");
```

Si può ottenere come valore restituito il numero di record su cui agisce l'istruzione.

Nota

Il parametro di *executeUpdate()* può essere anche una istruzione SQL *CREATE*.

Esempio 9.6

Classe *Database* con metodi per connettersi a un database con DSN *origine*, chiudere la connessione, visualizzare le righe della tabella *elenco*, inserire un record prestabilito nella tabella.

```
import java.sql.*;

public class Database {
    private Connection con;
    private Statement st;

    public void connetti() {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        }
```

Loescher Editore - Vietata la vendita e la diffusione

```
        catch(ClassNotFoundException e) {System.out.println(e.getMessage());}
        try {
            con = DriverManager.getConnection("jdbc:odbc:origine");
            st = con.createStatement();
        }
        catch(SQLException e) {System.out.println(e.getMessage());}
    }

    public void chiudi() {
        try {
            st.close();
            con.close();
        }
        catch(SQLException e) {System.out.println(e.getMessage());}
    }

    public void visualizza() {
        try {
            ResultSet rs = st.executeQuery("SELECT * FROM elenco");
            while(rs.next()) {
                String citta = rs.getString("citta");
                String cap = rs.getString("cap");
                System.out.println("Citta' : " + citta);
                System.out.println("Cap : " + cap);
            }
        }
        catch(SQLException e) {System.out.println(e.getMessage());}
    }

    public void inserisci() {
        try {
            st.executeUpdate("INSERT INTO elenco (citta, cap) VALUES ('Padova', '35100')");
        }
        catch(SQLException e) {System.out.println(e.getMessage());}
    }
}
```

```
public class ProvaDatabase {
    public static void main (String[] args) {
        Database db = new Database();
        db.connetti();
        db.visualizza();
        db.inserisci();
        System.out.println("Dopo l'inserimento:");
        db.visualizza();
        db.chiudi();
    }
}
```

Figura 9.14 *Esecuzione dell'esempio*

Loescher Editore - Vietata la vendita e la diffusione

Uso di parametri

Per usare dei **parametri** nelle istruzioni SQL, invece di un oggetto di tipo *Statement* si deve usare un oggetto di tipo ***PreparedStatement***.

PreparedStatement è una interfaccia che estende l'interfaccia *Statement*.

Si può creare un oggetto di tipo *PreparedStatement* usando il metodo ***prepareStatement()*** dell'interfaccia ***Connection***.
Il metodo *prepareStatement()* ha come argomento una stringa con una istruzione SQL in cui i parametri vanno indicati con dei punti di domanda.

Per **inserire i valori mancanti** si usano metodi specifici di *PreparedStatement* in base al tipo del parametro, che richiedono come argomenti l'indice del parametro che si vuole impostare e il valore da assegnare:

void **setDouble**(int parameterIndex, double x)	inserisce un valore *double*;
void **setInt**(int parameterIndex, int x)	inserisce un valore *int*;
void **setObject**(int parameterIndex, Object x)	inserisce un oggetto;
void **setString**(int parameterIndex, String x)	inserisce una stringa;
...	

Per **eseguire l'istruzione** si può usare uno dei metodi di *PreparedStatement*:

boolean **execute**()	permette di eseguire una qualsiasi istruzione SQL;
ResultSet **executeQuery**()	permette di eseguire operazioni di interrogazione (istruzioni *SELECT*) e restituisce un *ResultSet*;
int **executeUpdate**()	permette di eseguire operazioni di aggiornamento (istruzioni *INSERT*, *UPDATE* o *DELETE*) e restituisce il numero di record su cui ha effetto.

Esempio 9.7

Metodo per inserire nella tabella *elenco* un record i cui dati vengono passati come parametri.

```
public void inserisci(String pCitta, String pCap) {
    PreparedStatement pst;
    try {
        pst = con.prepareStatement("INSERT INTO elenco (citta, cap) VALUES (?, ?)");
        pst.setString(1, pCitta);
        pst.setString(2, pCap);
        pst.execute();
        pst.close();
    }
    catch(SQLException e) {
        System.out.println(e.getMessage());
    }
}
```

- - - - - - - - - - - - - -

```
public class ProvaDatabase {
    public static void main (String[] args) {
        Database db = new Database();
        db.connetti();
        db.inserisci("Rovigo", "45100");
```

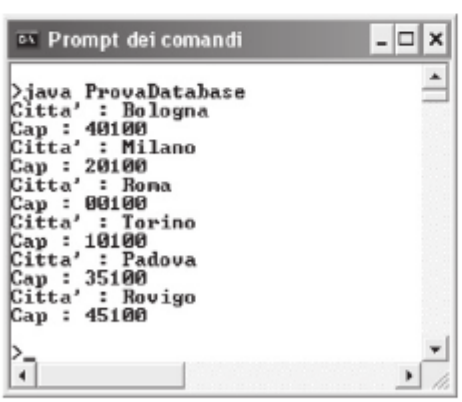


Figura 9.15 *Esecuzione dell'esempio*

Loescher Editore - Vietata la vendita e la diffusione

```
            db.visualizza();
            db.chiudi();
        }
    }
```

Paginazione e aggiornamento con i metodi di ResultSet

L'oggetto *ResultSet* creato di default ha un cursore che si muove solo in avanti e non può essere modificato.
Si può richiedere che l'oggetto *ResultSet* possa essere paginabile (*scrollable*) e modificabile (*updatable*).

Si può specificare il **tipo di *ResultSet*** da creare con i metodi *createStatement()* e *prepareStatement()* di *Connection*:

Statement **createStatement**(int resultSetType, int resultSetConcurrency)
PreparedStatement **prepareStatement**(String, sql, int resultSetType, int resultSetConcurrency)

resultSetType specifica il tipo di cursore di un *ResultSet*; può avere i valori:

ResultSet.TYPE_FORWARD_ONLY	indica un *ResultSet* il cui cursore si muove solo in avanti;
ResultSet.TYPE_SCROLL_INSENSITIVE	indica un *ResultSet* che è paginabile ma non sensibile ai cambiamenti fatti dagli altri;
ResultSet.TYPE_SCROLL_SENSITIVE	indica un *ResultSet* che è paginabile e sensibile ai cambiamenti fatti dagli altri;

resultSetConcurrency specifica il tipo di concorrenza di un *ResultSet*; può avere i valori:

ResultSet.CONCUR_READ_ONLY	indica un *ResultSet* che non può essere modificato;
ResultSet.CONCUR_UPDATABLE	indica un *ResultSet* che può essere modificato.

Metodi di ResultSet per la paginazione (scrolling)

boolean **absolute**(int row)	sposta il cursore alla riga specificata (restituisce *true* se il cursore si trova su una riga valida);
void **afterLast**()	sposta il cursore dopo l'ultima riga;
void **beforeFirst**()	sposta il cursore prima della prima riga;
boolean **first**()	sposta il cursore alla prima riga (restituisce *true* se il cursore si trova su una riga valida);
boolean **isAfterLast**()	verifica se il cursore si trova dopo l'ultima riga;
boolean **isBeforeFirst**()	verifica se il cursore si trova prima della prima riga;
boolean **isFirst**()	verifica se il cursore si trova sulla prima riga;
boolean **isLast**()	verifica se il cursore si trova sull'ultima riga;
boolean **last**()	sposta il cursore sull'ultima riga (restituisce *true* se il cursore si trova su una riga valida);
boolean **next**()	sposta il cursore alla riga successiva (restituisce *true* se il cursore si trova su una riga valida);
boolean **previous**()	sposta il cursore alla riga precedente (restituisce *true* se il cursore si trova su una riga valida);
boolean **relative**(int rows)	sposta il cursore avanti o indietro del numero di righe specificato (restituisce *true* se il cursore si trova su una riga valida).

Loescher Editore - Vietata la vendita e la diffusione

Metodi di ResultSet per l'aggiornamento

void **cancelRowUpdates**()	annulla le modifiche fatte alla riga corrente;
void **deleteRow**()	cancella la riga corrente dal database;
void **insertRow**()	inserisce il contenuto della riga di inserimento (insert row) nel database;
void **moveToCurrentRow**()	sposta il cursore alla riga corrente;
void **moveToInsertRow**()	sposta il cursore alla riga di inserimento (insert row);
void **refreshRow**()	ricarica la riga corrente dal database;
void **updateRow**()	aggiorna la riga corrente nel database;
void **updateDouble**(int columnIndex, double x)	aggiorna il valore in un campo *double* dato il numero della colonna;
void **updateDouble**(String columnName, double x)	aggiorna il valore in un campo *double* dato il nome del campo;
void **updateInt**(int columnIndex, int x)	aggiorna il valore in un campo *int* dato il numero della colonna;
void **updateInt**(String columnName, int x)	aggiorna il valore in un campo *int* dato il nome del campo;
void **updateObject**(int columnIndex, Object x)	aggiorna il valore in un campo *Object* dato il numero della colonna;
void **updateObject**(String columnName, Object x)	aggiorna il valore in un campo *Object* dato il nome del campo;
void **updateString**(int columnIndex, String x)	aggiorna il valore in un campo *String* dato il numero della colonna;
void **updateString**(String columnName, String x)	aggiorna il valore in un campo *String* dato il nome del campo.

Si può **modificare una riga** con le istruzioni:

rs.absolute(5);	sposta il cursore alla quinta riga;
rs.updateString("cap", "35100");	aggiorna la colonna *cap* della riga *5* al valore *35100*;
rs.updateRow();	aggiorna la riga nel database.

Un *ResultSet* modificabile ha associata una riga speciale di inserimento (**insert row**) per costruire righe da inserire.

Si può **inserire una nuova riga** con le istruzioni:

rs.moveToInsertRow();	sposta il cursore alla riga speciale di inserimento;
rs.updateString(1, "Rovigo");	prepara nella prima colonna il nome della città;
rs.update String(2, "45100");	prepara nella seconda colonna il CAP;
rs.insertRow();	inserisce la riga nel database;
rs.moveToCurrentRow();	sposta il cursore alla riga corrente.

I metadati

Si possono ottenere informazioni sullo schema del database (**metadati**) tramite le interfacce *DatabaseMetaData* e *ResultSetMetaData*.

L'interfaccia ***DatabaseMetaData*** fornisce dati generici come tipo di database, versione, istruzioni SQL supportate e così via.

ResultSetMetaData fornisce informazioni sulle colonne di un *ResultSet*, e quindi per esempio di una tabella.

Loescher Editore - Vietata la vendita e la diffusione

Si può ottenere un oggetto di tipo *ResultSetMetaData* con il metodo ***getMetaData()*** dell'interfaccia ***ResultSet***.

L'interfaccia *ResultSetMetaData* consente l'accesso ai metadati di un *ResultSet* tramite i suoi metodi:

int **getColumnCount**()	restituisce il numero delle colonne del *ResultSet*;
String **getColumnName**(int column)	restituisce il nome della colonna specificata;
String **getColumnTypeName**(int column)	restituisce il tipo della colonna specificata.

Esempio 9.8

Stampa del nome e del tipo delle colonne della tabella *elenco*.

```
public void struttura() {
    try {
        st = con.createStatement();
        ResultSet rs = st.executeQuery("SELECT * FROM elenco");
        ResultSetMetaData rsmd = rs.getMetaData();
        int numeroColonne = rsmd.getColumnCount();
        for (int i=1; i<=numeroColonne; i++) {
            String nome = rsmd.getColumnName(i);
            String tipo = rsmd.getColumnTypeName(i);
            System.out.println(nome + " tipo: " + tipo);
        }
        st.close();
    }
    catch(SQLException e) {System.out.println(e.getMessage());}
}
```

```
public class ProvaDatabase2 {
    public static void main (String[] args) {
        Database db = new Database();
        db.connetti();
        db.struttura();
        db.chiudi();
    }
}
```

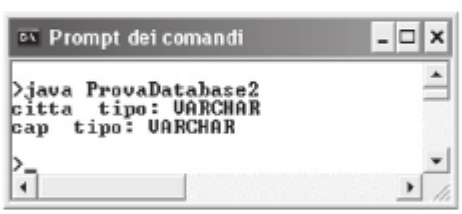


Figura 9.16
Visualizzazione della struttura del database

Usando i metadati si può scrivere un metodo per visualizzare nomi delle colonne e valori di una tabella qualsiasi.

Esempio 9.9

Lettura dei dati della tabella *elenco* (ha solo campi stringa) usando i metadati.

```
public void visualizza() {
        try {
        st = con.createStatement();
        ResultSet rs = st.executeQuery("SELECT * FROM elenco");
        ResultSetMetaData rsmd = rs.getMetaData();
        int numeroColonne = rsmd.getColumnCount();
        for (int i=1; i<=numeroColonne; i++) {
            System.out.print(rsmd.getColumnName(i) + "\t");
        }
```

Loescher Editore - Vietata la vendita e la diffusione

```
                System.out.println();
                while(rs.next()) {
                        for (int i=1; i<=numeroColonne; i++) {
                                System.out.print(rs.getString(i) + "\t");
                        }
                        System.out.println();
                }
                st.close();
        }
        catch(SQLException e) {
                System.out.println(e.getMessage());
        }
    }
}
```

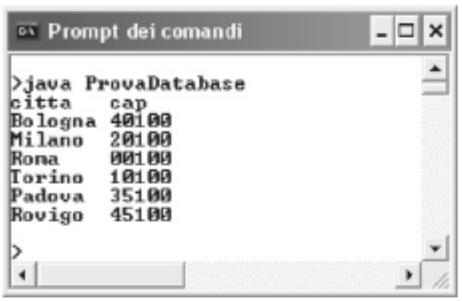


Figura 9.17 *Nomi delle colonne e valori della tabella elenco usando i metadati*

Se i campi sono di vario tipo bisogna testare il tipo di campo e, in base al tipo, leggere il valore con il metodo adatto.

VERIFICA

Questionario

1. Quali sono le principali tecnologie di accesso ai database?
2. Che cosa sono i driver per database?
3. Che cos'è OLE DB?
4. Quali sono gli oggetti principali di ADO?
5. Quali sono le possibili modalità di lavoro sui database con ADO.NET?
6. Che cos'è il data binding?
7. Che cosa sono i driver JDBC-ODBC?
8. In quale package si trovano le classi dell'API JDBC?
9. Come funziona un cursore?
10. Quali tipi di *ResultSet* si possono utilizzare e come si possono ottenere?
11. Che cosa sono i metadati?

Esercizi

1 ..... Visualizzare i record della tabella *elenco* dell'esempio 9.1 uno alla volta, con pulsanti di paginazione (avanti, indietro, inizio, fine).

2 ..... Utilizzare il data binding (come nell'esempio 9.4) per presentare i nomi della città in una casella di riepilogo.

3 ..... Visualizzare i record della tabella *elenco* dell'esempio 9.6 uno alla volta, con pulsanti di paginazione (avanti, indietro, inizio, fine).

4 ..... Trasformare l'esempio 9.8 in modo che possa stampare nome e tipo delle colonne di una tabella qualsiasi.

5 ..... Trasformare l'esempio 9.9 in modo che possa leggere i dati di una tabella qualsiasi, con campi stringa.

Loescher Editore - Vietata la vendita e la diffusione

6 ..... Realizzare un metodo per visualizzare nomi delle colonne e valori di una tabella qualsiasi, di qualsiasi tipo siano i campi.

PROPOSTE DI LAVORO

1 ..... Gestire un database con una tabella delle anagrafiche degli studenti di una classe (codice, nome, indirizzo, data di nascita ecc.); si deve poter:
- inserire i dati di uno studente;
- visualizzare i dati di tutti gli studenti;
- visualizzare i dati di uno studente richiesto, dato il codice o il nome;
- visualizzare i dati di tutti gli studenti il cui nome inizia con una data lettera;
- visualizzare l'elenco degli anni in cui sono nati gli studenti;
- modificare l'indirizzo di uno studente richiesto, dato il codice;
- cancellare i dati di uno studente, dato il codice.

2 ..... In un database *Calendario* di appuntamenti un appuntamento ha una data, un'ora di inizio e una di fine e una descrizione;
- aggiungere appuntamenti;
- eliminare appuntamenti annullati;
- stampare tutti gli appuntamenti;
- stampare l'elenco di appuntamenti per una giornata;
- stampare l'elenco di appuntamenti in un intervallo di ore.

Realizzare un'interfaccia grafica per la gestione delle varie operazioni.

3 ..... Realizzare un'interfaccia grafica per gestire un database di articoli con codice, descrizione e prezzo, con operazioni come:
- visualizzare i record in successione, uno alla volta;
- visualizzare un record dato il codice;
- visualizzare un record data la descrizione;
- modificare il prezzo di un articolo.

4 ..... Predisporre un'interfaccia grafica per gestire un database *Vocabolario* inglese/italiano e viceversa.

5 ..... In un database di geografia con stati e capitali, gestire:
- inserimento di dati;
- visualizzazione dei dati;
- ricerca della capitale, dato lo stato;
- ricerca dello stato, data la capitale.

Predisporre la gestione di domande su stati e capitali e il conteggio delle risposte. Prevedere la possibilità di fare una gara fra più persone.
Aggiungere al database le immagini delle bandiere degli stati e predisporre test anche sulle bandiere.

6 ..... Creare un motore di ricerca che indicizzi pagine Web fornite in input.
Si deve creare un elenco di termini da aggiornare man mano che in una pagina si trova un nuovo termine. Per ogni termine si deve mantenere l'elenco delle pagine in cui compare, il numero delle occorrenze in ciascuna pagina e la posizione in cui si trovano (titoli, link ecc.). Di ogni pagina si deve conoscere titolo, descrizione ed elenco delle parole chiave inserite nei metatag; si deve inoltre mantenere il numero dei link che puntano alla pagina.
Alla ricerca di un termine si devono fornire le pagine che lo contengono ordinate in base al numero di presenze del termine (dando maggior peso ai termini che si trovano nei titoli) e, per ugual numero, in base al numero dei link che puntano alla pagina.

Loescher Editore - Vietata la vendita e la diffusione

10 Accesso a database via Web

Prerequisiti

- Funzionamento del servizio Web e linguaggio HTML
- DBMS relazionali e linguaggio SQL
- Modalità e tecnologie di accesso ai database

Obiettivi

Conoscenze

- Protocollo HTTP e interfaccia CGI
- Concetti generali relativi alle applicazioni server side, alla pubblicazione di un sito e alla configurazione di un server Web
- PHP e MySQL
- JSP e JDBC
- ASP e ADO
- ASP.NET e ADO.NET
- Problemi di sicurezza delle applicazioni server side

Abilità/capacità

- Installare e configurare un server Web (Apache, Tomcat, IIS)
- Utilizzare diverse tecnologie per scrivere applicazioni server side che interagiscono con database

Loescher Editore - Vietata la vendita e la diffusione

10.1 IL PROTOCOLLO HTTP

Per consultare una pagina Web il browser stabilisce una connessione **TCP** con il server Web; dopo aver stabilito la connessione, il client invia una richiesta e il server risponde; poi la connessione viene chiusa; se la pagina contiene immagini, per ogni immagine viene ripetuto il procedimento, aprendo una nuova connessione.

Il protocollo **HTTP** (Hyper Text Transfer Protocol) definisce il formato delle **richieste** del browser e delle **risposte** del server, descrivendo in dettaglio le intestazioni e il corpo dei messaggi; ogni interazione tra client e server consiste in una sola richiesta e una sola risposta.

Il protocollo HTTP è **privo di stati**, cioè ad ogni connessione non ricorda cosa è avvenuto in precedenza; per mantenere informazioni ottenute nelle connessioni precedenti si possono usare i **cookie** (stringhe di testo memorizzate sul computer dell'utente).

Le richieste del **client** chiedono di trasferire un file di risorse (documento di testo, immagine, file audio o video ecc.) dal computer del server al computer del client.
Le richieste possono essere semplici o complete.

Le richieste contengono un metodo (GET o POST) seguito da un URL e possono contenere delle intestazioni che danno informazioni aggiuntive sulla richiesta, come:

From	indica l'indirizzo di posta elettronica del client;
If-Modified-Since	rende il metodo condizionale; chiede al server di trasferire il file solo se è stato modificato dopo una determinata data.

Il **server** per ogni richiesta manda un messaggio di risposta, poi chiude la connessione.
Il **messaggio di risposta** contiene un codice di stato a tre cifre e una descrizione esplicativa e può contenere delle intestazioni. I codici di stato usati più frequentemente sono *200 (OK)*, *400 (richiesta non corretta)*, *403 (accesso proibito)* e *404* (*risorsa non trovata*).

I campi di **intestazione** della risposta danno informazioni aggiuntive sulla risposta, per esempio:

Content-Length	lunghezza dei dati inviati;
Content-Type	tipo MIME dei dati inviati;
Last-Modified	usato per indicare la data dell'ultima modifica del file inviato;
Server	informazioni sul software del server usato per creare la risposta;
Set-Cookie	usato per memorizzare un cookie sul computer dell'utente.

10.2 L'INTERFACCIA CGI

L'**interfaccia CGI** (Common Gateway Interface) descrive come avviene lo scambio di dati tra il browser e le applicazioni server side.

Le applicazioni server side e l'interfaccia CGI

Le applicazioni server side sono applicazioni in esecuzione sul server Web che possono ricevere un input dall'utente e creare pagine Web di risposta dinamicamente.
L'utente normalmente inserisce i dati tramite un modulo in una pagina Web; i dati vengono inviati all'applicazione sul server che li elabora e restituisce un documento HTML con la risposta.

Richiesta di una pagina Web

- il browser chiede una pagina con una richiesta HTTP;
- il server invia una risposta HTTP.

Loescher Editore - Vietata la vendita e la diffusione

Se l'URL specificato nella richiesta del browser si riferisce a una **applicazione server side**:

- il server, prima di inviare la risposta, esegue l'applicazione;
- l'applicazione produce una risposta (in genere una pagina Web);
- il server invia la risposta.

Il browser può **inviare dati** all'applicazione server side:

- inserendoli in un modulo, che richiede l'esecuzione dell'applicazione con l'attributo *action*,
- accodandoli all'URL dell'applicazione, nel formato richiesto dall'interfaccia CGI.

Quando viene richiesta l'esecuzione di un'applicazione server side il server predispone i valori di opportune **variabili d'ambiente** che vengono usate dall'applicazione come input; le variabili d'ambiente rappresentano il modo con cui il server comunica con l'applicazione server side.

Le **variabili d'ambiente** contengono informazioni sul browser che effettua la richiesta, sul server che gestisce la richiesta e sui dati che vengono passati all'applicazione.

Alcune variabili d'ambiente

REQUEST_METHOD	metodo usato nella richiesta (*GET* o *POST*);
QUERY_STRING	dati inviati con il metodo *GET* (contiene le informazioni che seguono il punto interrogativo nell'URL che è stato specificato per richiedere l'applicazione);
CONTENT_LENGTH	quantità di dati inviati con il metodo *POST* disponibili sullo *stdin*;
DOCUMENT_ROOT	directory root del server;
HTTP_USER_AGENT	tipo di browser usato dall'utente;
REMOTE_ADDR	indirizzo IP del client che ha effettuato la richiesta;
REMOTE_HOST	nome di dominio del client che ha effettuato la richiesta;
SERVER_NAME	nome di dominio del server (o, se non disponibile, l'indirizzo IP);
SERVER_SOFTWARE	nome e versione del software del server Web;
SERVER_PORT	porta TCP a cui sono state inviate le intestazioni di richiesta (porta predefinita *80*).

Richiesta di un'applicazione server side con un modulo

Il tag ***<form>*** ha gli attributi:

action	URL dell'applicazione che deve essere eseguita sul server;
method	metodo usato per la richiesta HTTP; valori: *GET* (valore di default) o *POST*;

Vengono inviati all'applicazione server side, come coppie *nome=valore*, tutti i campi del modulo per cui è stato definito un nome con l'attributo ***name***. Il modulo viene inviato all'applicazione specificata nell'attributo *action* quando si fa clic su un pulsante.

Normalmente come **pulsante per inviare** il modulo si usa ***submit***; dando un nome al pulsante viene inviata la stringa *nome=valore*, dove *valore* è il valore dell'attributo *value*, cioè la scritta che compare sul pulsante.

Interfaccia CGI per la richiesta

Codifica dei dati

- I dati trasferiti sono codificati in coppie *nome=valore*;
- le coppie sono separate tra loro dal simbolo *&*;
- gli spazi nel valore vengono sostituiti dal simbolo *+*;

Loescher Editore - Vietata la vendita e la diffusione

- gli spazi negli URL e i caratteri speciali come segni di percentuale o punti di domanda vengono codificati; la codifica è costituita dal simbolo % seguito dal codice ASCII in esadecimale.

Tabella 10.1 *Codice ASCII di alcuni caratteri speciali*

spazio (nell'URL)	%20	<	%3C	>	%3E	"	%28	?	%3F
&	%26	/	%2F	=	%3D	#	%23	%	%25

Metodi GET e POST

Con il metodo ***GET*** i dati vengono inviati alla fine dell'URL indicato nell'attributo *action*, dopo un punto interrogativo (inserito automaticamente); sul server i dati vengono inseriti nella variabile d'ambiente *QUERY_STRING*.

I dati inviati sono visibili nella barra indirizzo, dopo l'URL e il punto interrogativo (anche i campi di tipo *password*). Inoltre la quantità dei dati che si possono trasmettere è limitata dalla lunghezza massima dell'URL che è di 255 caratteri.

Con il metodo ***POST*** i dati vengono inviati come entità in fondo alla richiesta; sul server i dati vengono resi disponibili sullo standard input dell'applicazione; la lunghezza dei dati viene inserita nella variabile d'ambiente *CONTENT_LENGTH*.

Richiesta di un'applicazione server side con un collegamento ipertestuale o un indirizzo nella barra indirizzi del browser

Si può richiamare un'applicazione server side anche con un **collegamento ipertestuale** o inserendone direttamente l'**URL nella barra indirizzi** del browser.

Il metodo usato nella richiesta HTTP in questi casi è sempre ***GET***.

Si possono passare dei dati all'applicazione codificandoli come coppie *nome=valore* e aggiungendoli dopo l'URL e un punto interrogativo (query string).

```
<a href="URLApplicazione?nome1=valore1&nome2=valore2">collegamento</a>
```

Riassumendo: metodi GET e POST

Il metodo usato nella richiesta HTTP può essere *GET* o *POST*.
Se l'URL dell'applicazione server side è un **collegamento ipertestuale** o un **indirizzo nella barra del browser** il metodo usato è sempre *GET*.
Se l'URL è specificato nell'**attributo *action* di un modulo** il metodo può essere *GET* o *POST* secondo il valore dell'attributo *method* (se non è presente, il valore di default è *GET*).

Interfaccia CGI per la risposta

L'applicazione server side trasmette la risposta al server sullo **standard output**.

Normalmente l'applicazione server side scrive sullo standard output:

- l'intestazione: *Content-type: text/html*, che indica il tipo di dati inviato al browser;
- una riga vuota;
- il documento HTML generato.

Il browser interpreta il documento HTML e visualizza la pagina.

Loescher Editore - Vietata la vendita e la diffusione

10.3 APPLICAZIONI SERVER SIDE

Tecnologie server side

Le più diffuse tecnologie server side sono **PHP** (HyperText Preprocessor), **ASP** (Active Server Pages) e **JSP** (JavaServer Pages).
Queste tecnologie permettono di inserire codice in un linguaggio di **scripting** in una normale pagina HTML.
Il codice viene eseguito **sul server**, prima che la pagina sia inviata al browser.

PHP è una tecnologie freeware, open source e multipiattaforma; inoltre è facile da imparare.
ASP è una tecnologia Microsoft quindi si può usare solo su piattaforma *Windows* ma è molto diffusa; si è evoluta nella tecnologia **ASP.NET**.
JSP è una tecnologia che utilizza **Java**, come i servlet. I **servlet** sono classi Java che vengono compilate per produrre file *.class* eseguiti sul server; l'uso dei servlet richiede una buona conoscenza della programmazione Java. JSP permette di realizzare applicazioni server side in Java in modo semplice: infatti le pagine JSP vengono tradotte automaticamente in servlet dal server Web. Servlet e JSP sono tecnologie freeware e multipiattaforma, portabili, orientate agli oggetti, che permettono di riutilizzare il codice (grazie anche all'uso dei JavaBeans).
Una tecnologia basata su **XML** è **ColdFusion**.

Le applicazioni server side non hanno interfaccia grafica. L'interfaccia grafica è la pagina Web. Le applicazioni server side possono ricevere dati dal browser e possono interagire con database o con documenti XML.

Dato che il protocollo HTTP è **privo di stati**, per mantenere informazioni ottenute nelle connessioni precedenti le applicazioni server side possono usare i **cookie** o gestire le **sessioni**.

Le applicazioni server side possono essere richiamate inserendo l'URL nella barra indirizzi del browser, oppure da un collegamento ipertestuale in una pagina HTML o da un modulo.

Cookie

Un **cookie** è una stringa di testo memorizzata sul computer dell'utente che permette di mantenere informazioni ottenute nelle connessioni precedenti.

I cookie possono servire per memorizzare delle informazioni tra una richiesta e l'altra dato che il protocollo HTTP è privo di stati.
I cookie essendo solo stringhe di testo non sono pericolosi per la sicurezza ma possono essere un pericolo per la privacy.
Il browser può essere configurato in modo da rifiutare i cookie o in modo che sia richiesto per ogni cookie se accettarlo o meno; si possono anche abilitare i cookie solo per il sito a cui si è connessi.
I cookie in genere sono grandi 50-150 byte e non possono avere dimensioni superiori a 4 KB (dimensione massima dell'intestazione di cookie nel protocollo HTTP).
Un server può memorizzare al massimo 20 cookie sul computer dell'utente; il browser può memorizzare in tutto al massimo 300 cookie.
Internet Explorer memorizza i cookie nella directory *Temporary Internet Files* di ogni utente.

I cookie vengono inviati al browser con l'intestazione di risposta HTTP *Set-cookie*.

L'**intestazione *Set-cookie***, contiene:

- una coppia *nome=valore*;
- una data di scadenza: indica al browser quando scade il cookie, cioè da quando non è più valido;

Loescher Editore - Vietata la vendita e la diffusione

- un dominio: indica il dominio a cui il cookie può essere inviato (non può essere diverso da quello che ha richiesto la creazione del cookie);
- un percorso: indica il path del sito Web a cui il cookie può essere inviato (se non è specificato è il path corrente).

Se il browser è configurato in modo da permettere l'uso dei cookie, i cookie inviati rimangono memorizzati fino alla scadenza. Quando l'utente naviga su una pagina che corrisponde al dominio e al percorso di un cookie memorizzato, il browser invia il cookie al server.

Una scadenza di zero secondi fa in modo che il cookie sia valido solo per la durata della sessione dell'utente. Quando il browser viene chiuso il cookie non è più valido.

Sessioni

Una **sessione** è l'attività svolta dall'utente dalla prima volta in cui accede a una pagina dell'applicazione, fino a quando chiude il browser (comprende tutte le pagine aperte nella stessa finestra del browser).

Nelle tecnologie server side una sessione è un oggetto che fornisce un'associazione tra client e server che persiste su connessioni multiple, mantenendo informazioni di stato.

Ogni sessione è identificata da un **ID** univoco.

Per gestire la sessione viene memorizzato un **cookie** sul computer dell'utente che contiene l'ID di sessione. Il browser deve essere configurato in modo da accettare i cookie. Se l'utente disabilita i cookie nel browser le sessioni non funzionano.

Per poter usare le sessioni anche se l'utente disabilita i cookie nel browser, si può passare l'ID di sessione da una pagina all'altra mediante gli URL codificati (URL che includono l'ID di sessione).

Nota

Le tecnologie server side permettono di **gestire le sessioni** e di memorizzare **attributi di sessione** (cioè valori validi per la durata della sessione).

Architettura 3-tier

Le applicazioni server side vengono spesso utilizzate per eseguire elaborazioni su database. In genere l'accesso a DBMS utilizza un'**architettura 3-tier** (a tre livelli) composta da:

- **browser Web**: livello di presentazione che ha il compito di gestire l'interfaccia;
- **pagine server side** eseguite sul server: costruiscono i comandi SQL da passare al database e preparano le pagine Web da mandare al browser;
- **DBMS**: esegue i comandi SQL sul database e restituisce i risultati alla pagina.

I tre livelli possono trovarsi indifferentemente sullo stesso computer o su tre computer diversi.

Le applicazioni Web possono avere anche un'architettura con più di tre livelli chiamata genericamente n-tier.

Nota

Le applicazioni server side possono accedere a un database tramite un **driver ODBC** che permette di interagire con il database utilizzando istruzioni SQL.

Loescher Editore - Vietata la vendita e la diffusione

Pubblicazione di un sito

Il sito deve essere **pubblicato** su un server Web.
Un **server Web** è un programma che permette la consultazione di pagine Web memorizzate in opportune cartelle.
Se il sito deve essere reso disponibile solo all'interno di una intranet basta collocare le pagine sul server, nella cartella appropriata.
Se il sito deve essere reso disponibile su tutta la rete Internet bisogna rendere accessibile il server Web dall'esterno o utilizzare un servizio offerto da un provider; le possibilità sono il servizio di hosting o il servizio di housing.
L'**hosting** è un servizio che ospita più siti Web su una singola macchina del provider.
L'**housing** prevede un server di proprietà dell'utente, gestito dall'utente, collocato direttamente nei locali del provider.

L'housing è costoso e richiede esperienza nella gestione del server; è adatto per siti con moltissimo traffico; il vantaggio è che potendo amministrare direttamente il server si possono scegliere tutti i componenti desiderati: sistema operativo, server Web, tecnologie server side come ASP o PHP, database e così via. I servizi di hosting hanno costi minori, ma anche prestazioni minori (molti siti condividono lo stesso computer e la stessa banda).

Si possono pubblicare le pagine nello spazio Web usando il servizio **FTP**.
Nel client FTP bisogna indicare **l'indirizzo del server FTP** e **l'account** da usare per il collegamento.

Registrazione di un nome di dominio

Per poter avere un nome di dominio del tipo *www.nomeScelto.dominio* bisogna **registrare un nome di dominio** di secondo livello (*nomeScelto.dominio*).
La registrazione dei nomi di dominio nel dominio *it* è gestita dalla Registration Authority italiana (*www.nic.it*).
La registrazione dei nomi di dominio può essere fatta in modo più semplice appoggiandosi a un provider che provvede alla registrazione.

Registrazione nei motori di ricerca

Dopo aver creato il sito, per fare in modo che gli altri possano visitarlo, bisogna **registrare il sito** presso i principali **motori di ricerca**.
I motori di ricerca di solito contengono nella pagina principale un collegamento a una pagina con un modulo per sottoporre il proprio sito.

SERVER WEB

Per offrire un servizio Web su Internet o in una intranet si può usare un programma di gestione server Web.
I principali server Web sono *Apache* in ambiente *Linux* (ma anche *Windows*) e *Internet Information Server* su *Windows*.
Dal tipo di server dipendono prestazioni, affidabilità e sicurezza.
Per esempio può variare il numero di browser che possono accedere contemporaneamente al sito e la velocità di risposta.
La funzione di gestione dei server virtuali permette di gestire più siti Web su un unico server.
La **configurazione** di un server Web richiede tra l'altro di indicare in quale directory memorizzare le pagine Web e che tipo di accesso è possibile usare.
L'**accesso** alle pagine Web può essere **anonimo** oppure in base ad **autenticazione** (in tal caso al primo accesso del browser compare una finestra di dialogo che richiede account e password).

Loescher Editore - Vietata la vendita e la diffusione

DNS

Per permettere di accedere al sito usando un nome di dominio bisogna configurare anche il servizio **DNS**.

FTP per la pubblicazione delle pagine

Per rendere possibile la pubblicazione delle pagine Web mediante FTP bisogna installare e configurare un server FTP sullo stesso computer su cui è installato il server Web.
Un **server FTP** permette ai client di effettuare operazioni di upload o download di file.
Il server FTP può essere configurato in modo da permettere accessi anonimi o con account.
In base all'accesso e all'account usato possono essere limitate le operazioni permesse.
Per usare il server FTP per la pubblicazione delle pagine Web sul server Web, le directory di upload devono essere le directory usate dal server Web per la pubblicazione delle pagine.

Server Web accessibile dall'esterno

Se la rete locale è collegata a Internet e il server Web deve essere accessibile anche dall'esterno, il server Web deve avere un **indirizzo IP univoco** oppure deve essere configurato un router **NAT/PAT** che renda accessibile il server.
Per poter accedere al server con un nome di dominio, invece che solo mediante l'indirizzo IP, il server Web deve avere un **nome di dominio registrato** (in modo che possa essere individuato attraverso il DNS).
Se l'indirizzo IP è dinamico bisogna utilizzare un servizio **DDNS** (DNS dinamico).
Spesso i server pubblici vengono inseriti in una DMZ (demilitarized zone, o zona demilitarizzata) per motivi di sicurezza.

Avvio del server Web

Il server Web deve essere avviato manualmente, se non è configurato in modo da essere avviato automaticamente all'accensione del computer.

Accesso alle pagine Web

Se il server Web si trova sul computer locale, per accedere dal browser alle pagine pubblicate basta specificare l'indirizzo *http://localhost* (o *http://127.0.0.1*); se il server è impostato per usare una porta diversa dalla porta 80 predefinita bisogna specificare anche il numero di porta dopo il simbolo : (per esempio se la porta è 8080 *http://localhost:8080* o *http://127.0.0.1:8080*).
Se il server Web si trova su un altro computer della rete bisogna specificare al posto di 127.0.0.1 l'indirizzo IP di tale computer.

CONFIGURAZIONE DI APACHE

Apache è uno dei **server Web** più usati. Apache può essere configurato da interfaccia grafica o modificando direttamente il file di configurazione ***httpd.conf*** che in *Windows* si trova nella cartella *conf* all'interno della cartella di installazione di Apache e in Linux in */etc/httpd/conf*.

Loescher Editore - Vietata la vendita e la diffusione

In alcune distribuzioni di *Linux* Debian-based (Debian, Ubuntu..) i file di configurazione si trovano in */etc/apache2/*; il file di configurazione principale è *apache2.conf*, ma alcune impostazioni si trovano anche nei file *envvars* (variabili d'ambiente), *ports.conf* (porta usata dal server) e *sites-available/* (virtual host, usati in queste distribuzioni anche per indicare la directory radice dei documenti).

Nota

Il file di configurazione può contenere **direttive**, **blocchi**, che raggruppano un insieme di direttive che agiscono su specifiche directory o file, e **commenti** (che iniziano con il simbolo #).

Alcune direttive sono:

Port	specifica la porta corrispondente al servizio *http*;
DocumentRoot	specifica la directory in cui devono essere collocate le pagine Web;
DirectoryIndex	indica il file di default da restituire quando si accede a una directory invece che a un file specifico;
LoadModule	permette di caricare un modulo (serve per caricare l'interprete PHP).

Per usare PHP e MySQL bisogna:

- installare PHP e caricare il modulo dell'interprete PHP nel file di configurazione di Apache;
- installare MySQL e modificare il file di configurazione di PHP *php.ini* inserendo la direttiva *extension=php_mysql.dll* per Windows e *extension=mysql.so* per Linux, che carica effettivamente il modulo che permette l'interazione tra PHP e MySQL.

Di solito la direttiva *extension* è presente ma commentata; per abilitarla è sufficiente rimuovere il ";" iniziale. In PHP4 la direttiva era abilitata per default.

Nota

Xampp

Xampp è uno strumento che permette di ottenere una configurazione Apache-MySQL-PHP senza necessità di configurazione.
Per esempio per usare la versione "archivio ZIP" di XamppLite per *Windows* è sufficiente estrarre l'archivio in *C:* (viene creata la directory *C:\xampplite* in cui si trovano gli script *xampp_start.exe* per l'avvio di Xampp e *xampp_stop.exe* per lo spegnimento; la directory radice *DocumentRoot* è *C:\xampplite\htdocs*).

10.4 PHP

Configurazione

Si può personalizzare il comportamento del motore PHP modificando il file di configurazione ***php.ini***.

Il file *php.ini* si trova nella directory di installazione di PHP in *Windows* e in */etc/php/* o */etc/php5/apache2* in *Linux*.

Nota

Il file *php.ini* contiene direttive del tipo *nome = valore*.
Per abilitare una direttiva si usa uno dei valori *On*, *True* o *Yes*, per disabilitarla uno dei valori *Off*, *False* o *No*.

La funzione ***phpinfo()*** restituisce una tabella con informazioni sulla configurazione di PHP.

Loescher Editore - Vietata la vendita e la diffusione

Esempio 10.1

Esecuzione della funzione *phpinfo()*.

```
<html>
<head>
<title>PHP</title>
</head>
<body>
<?php phpinfo(); ?>
</body>
</html>
```

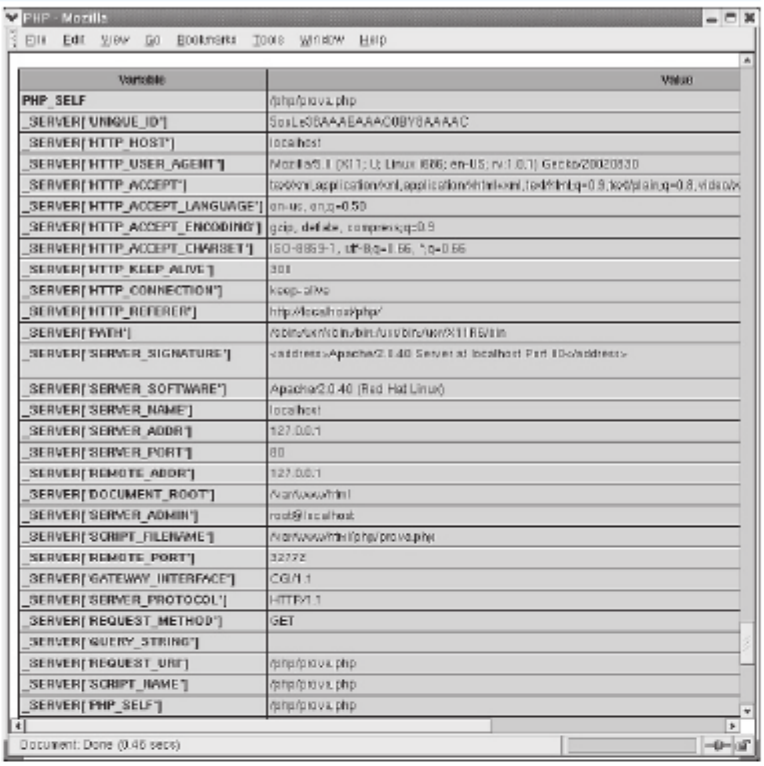


Figura 10.1 *Esecuzione dell'esempio: variabili d'ambiente*

Primi elementi

Il codice va inserito in una pagina HTML racchiuso tra i tag ***<?php*** e ***?>***. Le pagine che contengono codice PHP di solito hanno estensione ***.php***.

Esempio 10.2

Generazione di una pagina Web.

```
<html>
<head><title>PHP</title></head>
<body>
<?php
      print("<h1>PHP</h1>");
      print("Prima pagina")
?>
</body>
</html>
```

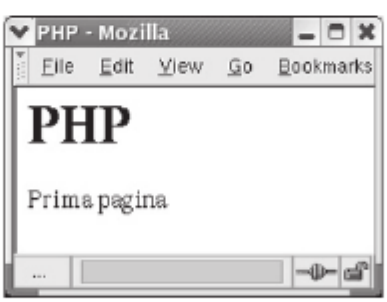


Figura 10.2 *Esecuzione dell'esempio*

L'istruzione *print("stringa")* scrive l'argomento nella pagina HTML.

Si può scrivere indifferentemente:

```
print("stringa")
print "stringa"
echo("stringa")
echo "stringa"
```

Loescher Editore - Vietata la vendita e la diffusione

Invece dello **stile canonico** che usa i tag *<?php* e *?>* si può usare:

lo **stile HTML**: usa i tag *<script language="php">* e *</script>*,
lo **stile breve**: usa i tag *<?* e *?>* (nel file di configurazione *php.ini* deve essere presente la direttiva *short_open_tag = On*);
lo **stile ASP**: usa i tag *<%* e *%>* (nel file di configurazione *php.ini* deve essere presente la direttiva *asp_tags = On*).

I **commenti** possono essere su una sola riga che inizia con // o # o su più righe racchiusi tra /* e */.

Variabili

I **nomi** delle variabili sono preceduti dal simbolo ***$*** e possono contenere lettere, numeri e il trattino di sottolineatura (ma non devono iniziare con un numero); è importante la distinzione tra maiuscolo e minuscolo (case sensitive).
Non è necessario dichiarare le variabili prima di usarle; per usare una variabile basta assegnarle un valore.
Si può **assegnare un valore** a una variabile con l'operatore di assegnazione =.
La funzione ***isSet()*** permette di verificare se a una variabile è stato assegnato o no un valore.

Le variabili in PHP non sono di un tipo particolare (si dice che PHP è **debolmente tipizzato**); a una variabile cioè è possibile assegnare un valore di qualsiasi tipo, e valori diversi in momenti diversi.
I **tipi di dati** disponibili sono *Integer*, *Double*, *String*, *Boolean*, *Array* e *Object*.
Alcune **conversioni di tipo** sono eseguite automaticamente in modo implicito in base al contesto.

Nell'istruzione ***print()*** le variabili indicate tra virgolette vengono interpretate (cioè viene stampato il valore e non il nome della variabile; per stampare il nome della variabile bisogna far precedere il simbolo $ dal carattere di escape (\) o racchiudere il nome tra apici (').

```
$variabile = 10;
$variabile = "ciao";
print("\$variabile = $variabile <br/>");  // stampa $variabile = ciao
```

Costanti

Si può definire una costante con la funzione ***define()*** che ha come argomenti il nome della costante e il valore da assegnarle; una volta definita, la costante non può essere modificata.
Per convenzione il nome delle costante va espresso in maiuscole; per accedere al valore della costante non si usa il simbolo di dollaro ($).

```
define(COSTANTE, 50);
print("COSTANTE = " . COSTANTE);  // stampa COSTANTE = 50
```

Operatori

Operatori aritmetici	Operatori relazionali	Operatori logici
+ addizione	== uguaglianza	&& and And
- sottrazione	!= disuguaglianza	|| or Or
* moltiplicazione	=== identità (sono uguali e dello stesso tipo)	! Not
/ divisione		
% modulo (resto della divisione)	< minore	
+= -= *= /= %=	> maggiore	
	<= minore o uguale	
++ incremento	>= maggiore o uguale	
-- decremento		

Loescher Editore - Vietata la vendita e la diffusione

Operatore di concatenazione di stringhe

.	concatenazione
.=	*variabile .= espressione* corrisponde a *variabile = variabile . espressione*

L'operatore . concatena due stringhe (considera entrambi gli operandi come stringhe)

```
$stringa = "Ciao " . " mondo"  // valore  "Ciao mondo"
```

Operatore condizionale

?:	operatore condizionale

L'operatore condizionale ?: ha il significato *condizione ? valore per vero : valore per falso.*

Istruzioni di controllo

Ogni istruzione termina con un punto e virgola; blocchi di istruzioni vanno racchiusi tra parentesi graffe.

Istruzioni di selezione

```
if (condizione) {
        //istruzioni
}
```

```
if (condizione) {
        // istruzioni
}
else {
        // istruzioni
}
```

Selezione multipla

```
if (condizione) {
        // istruzioni
}
elseif (condizione2) {
        // istruzioni
}
else {
        // istruzioni
}
```

```
switch (espressione) {
        case valore1: //istruzione;
                break;
        case valore2: //istruzione;
                break;
        ...
        default: // istruzione;
}
```

Istruzioni cicliche

```
while (condizione) {
        // istruzione
}
```

Il ciclo viene ripetuto mentre la condizione è vera.

```
do {
        // istruzione
} while (condizione)
```

Il ciclo viene ripetuto mentre la condizione è vera; le istruzioni vengono sempre eseguite almeno una volta.

o in generale:

```
for (var i=inizio; i<=fine, i++) {
        // istruzioni
}
```

```
for (inizializzazione; condizione; aggiornamento) {
        // istruzioni
}
```

Loescher Editore - Vietata la vendita e la diffusione

Funzioni

Si può **dichiarare** una funzione con:

```
function nome() {
    // istruzioni
}
```

Si **richiama** la funzione con:

```
nome()
```

Le **variabili** definite all'interno della funzione sono **locali.**
All'interno delle parentesi può comparire un elenco di **parametri** separati da virgole.
Il passaggio dei parametri avviene **per valore**. Si può passare una variabile per riferimento facendola precedere dal simbolo *&*.

La funzione può restituire un valore mediante l'istruzione:

```
return espressione
```

```
function somma($primo, $secondo) {
    return $primo + $secondo;
}
print somma(3, 7);
```

Array indicizzati numericamente

Un array è un gruppo di elementi, anche di tipo diverso tra di loro.

Si può fare riferimento a un **elemento** dell'array con:

```
nomeArray[]
```

che si riferisce al primo elemento libero dell'array o con:

```
nomeArray[indice]
```

l'indice parte da 0.

La funzione ***count()*** dà il numero di elementi di un array.

```
$mesi[0] = "gennaio";
$mesi[1] = "febbraio";
$mesi[] = "marzo";
print($mesi[0]."<br/>");
print("Mesi inseriti: ".count($mesi)."<br/>");
```

Si può usare un ciclo *foreach* per scandire tutti gli elementi di un array. Ad ogni ciclo viene assegnato alla variabile il valore di un elemento dell'array.

```
foreach ($array as $value) {
    // istruzioni;
}
```

```
foreach ($mesi as $nome) {
    print($nome."<br/>");
}
```

Loescher Editore - Vietata la vendita e la diffusione

Array associativi

Gli array associativi usano come indici delle stringhe.

```
$giorni["gennaio"] = 31;
$giorni["febbraio"] = 28;
print($giorni["gennaio"]);
```

Si può usare un ciclo ***foreach*** per scandire tutti gli elementi di un array associativo. Ad ogni ciclo vengono assegnati alle variabili l'indice (chiave) e il valore di un elemento dell'array.

```
foreach ($array as $chiave => $valore) {
        // istruzioni;
}
```

```
foreach ($giorni as $mese => $numero ) {
        print($mese." = ".$numero."<br/>");
}
```

Nota

Gli array multidimensionali sono supportati come array di array (anche associativi).

Alcune funzioni per operare sugli array

array_merge()	fa il merge di due o più array;
array_push()	aggiunge elementi a un array;
array_shift()	elimina il primo elemento di un array;
sort()	ordina un array indicizzato numericamente;
asort()	ordina un array associativo in base ai valori;
ksort()	ordina un array associativo in base alle chiavi;
in_array()	verifica se un valore è presente in un array;
usort(), ***uasort()*** e ***uksort()***	ordinano un array (primo argomento) in base all'ordinamento specificato dalla funzione indicata come secondo argomento (la funzione deve restituire *0* se gli elementi sono uguali, *-1* se il primo è minore del secondo e *1* se è maggiore).

Stringhe

Le stringhe sono array di caratteri (indicizzati a partire da 0); si può accedere a un carattere come a un elemento dell'array.

```
print $stringa[0];
stampa il primo carattere della stringa
```

Alcune funzioni per operare sulle stringhe

strlen()	dà la lunghezza di una stringa;
trim()	elimina i caratteri bianchi all'inizio e alla fine di una riga;
strtoupper()	converte la stringa in maiuscolo;
strtolower()	converte la stringa in minuscolo;
strpos()	trova la posizione nella stringa specificata come primo argomento del carattere o della sottostringa specificati come secondo argomento;
substr()	restituisce una sottostringa di una stringa; ha come argomenti la stringa, la posizione da cui inizia la sottostringa ed opzionalmente la lunghezza della sottostringa.

Loescher Editore - Vietata la vendita e la diffusione

Gestione dei dati inviati dal browser

Il browser può **inviare dati** all'applicazione server side:

- inserendoli in un modulo che richiede l'esecuzione dell'applicazione con l'attributo *action*;
- accodandoli all'URL dell'applicazione, nel formato richiesto dall'interfaccia CGI.

Si può fare riferimento ai dati della query string (dati passati con il metodo *GET*) mediante l'array superglobale ***$_GET***.
Si può fare riferimento ai dati passati con il metodo *POST* mediante l'array superglobale ***$_POST***.

Gli **array superglobali** sono array associativi automaticamente globali ovunque.
Gli array *$_GET* e *$_POST* contengono le coppie nome/valore inviate rispettivamente col metodo *GET* o col metodo *POST*.
Altri array superglobali sono ***$_SERVER*** che permette di fare riferimento alle variabili d'ambiente, ***$_COOKIE*** che permette di fare riferimento ai cookie e ***$_SESSION*** che permette di fare riferimento alle variabili di sessione.

Nota

Nelle versioni precedenti di PHP si usavano gli array *$HTTP_GET_VARS*, *$HTTP_POST_VARS*, *$HTTP_SERVER_VARS*, *$HTTP_COOKIE_VARS* e *$HTTP_SESSION_VARS*.

Per accedere alle informazioni passate con la richiesta HTTP con il metodo *GET* si usa *$_GET['nome']*, dove *nome* è il nome di una coppia nome/valore; in pratica, se i dati sono passati mediante un modulo, è il nome di un elemento del modulo.

Esempio 10.3

Stampa dei valori dei parametri passati con la richiesta.

```
<html>
<head><title>PHP</title></head>
<body>
<h1>Parametri della richiesta</h1>
<?php
	print("Nome: ".$_GET['nome']."<br/>");
	print("Cognome: ".$_GET['cognome']);
?>
</body>
</html>
```

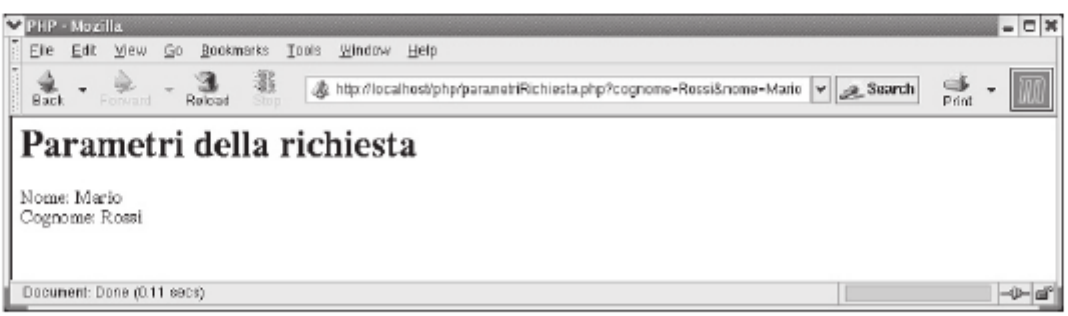


Figura 10.3 *Esecuzione dell'esempio*

Loescher Editore - Vietata la vendita e la diffusione

Modulo di richiesta

```
<html>
<head><title>PHP</title></head>
<body>
<form method="get" action="parametriRichiesta.php" ">
<table>
<tr><td>Cognome</td>
<td><input type="text" name="cognome"><td/></tr>
<tr><td>Nome</td>
<td><input type="text" name="nome"><td/></tr>
</table>
<br/>
<input type="submit" value="Invia!">
</form>
</body>
</html>
```

Figura 10.4
Esecuzione dell'esempio con i dati passati dal modulo

Se si usa il metodo *POST*, per accedere alle informazioni si usa *$_POST['nome']*, dove *nome* è il nome di un elemento del modulo.

Se nel file *php.ini* si inserisce la direttiva *register_globals = On*, i valori ottenuti con una richiesta HTTP vengono resi disponibili come variabili globali.
In pratica, se i dati sono passati mediante un modulo, per ogni elemento del modulo si ha una variabile globale che ha lo stesso nome dell'elemento (*$nome*, se *nome* è il nome dell'elemento del modulo).
Alle variabili globali si può accedere anche mediante l'array associativo *$GLOBALS* (*$GLOBALS['nome']*).
Queste tecniche però sono poco sicure e l'opzione *register_globals* è per default disattivata (e in futuro potrebbe non essere più supportata).

$_SERVER['PHP_SELF']

Il codice per la definizione del modulo e quello di analisi del modulo possono essere riuniti in un unica pagina; in tal caso lo script deve richiamare se stesso nell'attributo *action* del tag *form*.
Uno script può fare riferimento a se stesso mediante ***$_SERVER['PHP_SELF']***.
$_SERVER['PHP_SELF'] rappresenta il path dello script in esecuzione.

Nota
In precedenza si usava la variabile *$PHP_SELF*.

Nota
Quando il codice per la definizione del modulo e quello di analisi sono riuniti in un unica pagina, si usa la funzione *isset()* per verificare se i campi del modulo sono stati inizializzati, cioè se la chiamata dallo script avviene dal modulo oppure no. Le variabili inizializzate in seguito all'invio di un modulo vengono riconosciute sempre come impostate, anche se l'utente non ha inserito dati.

Loescher Editore - Vietata la vendita e la diffusione

Cookie

Per impostare un cookie si usa la funzione ***Setcookie()*** che ha come argomenti il nome del cookie, il valore, una data di scadenza (come timestamp), un dominio e un intero che deve valere *1* se il cookie deve essere inviato solo su connessioni sicure.
La funzione *Setcookie()*, deve essere chiamata prima di qualsiasi output sul browser.

Per leggere i cookie si può usare l'array superglobale ***$_COOKIE*** (*$_COOKIE['nomecookie']*).

Esempio 10.4

Uso dei cookie.

Memorizzazione di cookie

```
<?php
        Setcookie('colore','rosso',0);
        Setcookie('fiore','garofano',time()+3600)
?>
<html>
<head><title><PHP></title></head>
<body>
<h1>Cookie: memorizzazione</h1>
</body>
</html>
```

Setcookie("colore","rosso",0) imposta un cookie per la durata della sessione;
Setcookie('fiore','garofano',time()+3600) imposta un cookie che scade dopo un'ora.

Visualizzazione dei cookie

```
<html>
<head><title>PHP</title></head>
<body>
<h1>Cookie: visualizzazione</h1>
<?php
        print("colore: ".$_COOKIE['colore']."<br/>");
        print("fiore: ".$_COOKIE['fiore']);
?>
</body>
</html>
```

Figura 10.5 *Esecuzione dell'esempio*

Nota

In precedenza si usava $HTTP_COOKIE_VARS; se è impostata l'opzione *register_globals* nel file *php.ini*, viene creata una variabile globale col nome di ciascun cookie.

Sessioni

Ogni script PHP che utilizza le sessioni deve iniziare con una chiamata alla funzione ***session_start()*** (dato che imposta un cookie deve essere richiamata prima dell'invio di qualsiasi altro output al browser).
Se è disponibile un identificatore di sessione (in un cookie), vengono rese disponibili le variabili di sessione registrate in precedenza, altrimenti viene generato un nuovo ID di sessione.
Si può ottenere l'ID di sessione tramite la funzione ***Session_id()***.
Si possono gestire le variabili di sessione mediante l'array superglobale ***$_SESSION***.

Loescher Editore - Vietata la vendita e la diffusione

Esempio 10.5

Uso delle variabili di sessione.

Memorizzazione di alcune variabili di sessione

```
<?php
session_start();
?>
<html>
<head><title>PHP</title></head>
<body>
<h1>Sessioni: memorizzazione valori</h1>
<?php
        $_SESSION['utente'] = "Rossi";
        $_SESSION['password'] = "S&gr&t0";
?>
</body>
</html>
```

Visualizzazione delle variabili di sessione

```
<?php
session_start();
?>
<html>
<head><title>PHP</title></head>
<body>
<h1>Sessioni: visualizzazione valori</h1>
<?php
        print("L'ID di sessione e': ".Session_id()."<br/>");
?>
<p>I valori di sessione sono:</p>
<?php
        print("Utente: ".$_SESSION['utente']."<br/>");
        print("Password: ".$_SESSION['password']."<br/>");
?>
</body>
</html>
```

Figura 10.6 *Esecuzione dell'esempio*

10.5 GESTIONE DI DATABASE IN PHP (CON MYSQL)

Collegamento al server

Per prima cosa si deve richiamare la funzione ***mysql_connect()*** per aprire una connessione con il server di database.

mysql_connect*(host, utente, password)*

apre una connessione al server specificando l'host, un nome utente e una password con cui accedere al database; nessun parametro è obbligatorio: per default viene usato localhost come nome host. Se la connessione riesce, restituisce un identificativo di connessione da usare per le operazioni successive.

Loescher Editore - Vietata la vendita e la diffusione

Dopo aver stabilito una connessione con il server si deve scegliere un database.
Per selezionare un database si usa la funzione ***mysql_select_db()***.

mysql_select_db*(database)*	seleziona il database specificato; restituisce *true* se l'accesso al database ha esito positivo.

Al termine bisogna chiudere la connessione con la funzione ***mysql_close()***.

mysql_close*(connessione)*	chiude la connessione specificata.

Le istruzioni generali per collegarsi a un database quindi sono:

```
$conn = mysql_connect($host, $utente, $password);
if (!$conn) die('Impossibile connettersi: '.mysql_error());
mysql_select_db($database) or die ('Impossibile aprire il database: '.mysql_error());
....
mysql_close($conn)
```

Nota

La funzione *die()* termina lo script e visualizza il messaggio specificato.

Per realizzare un'operazione sul database si scrive un comando SQL e lo si passa come argomento alla funzione ***mysql_query()***.

mysql_query*(query)*	esegue la query e restituisce *true* se la query è stata eseguita con successo.

Visualizzazione di dati

Per visualizzare i dati di una tabella del database bisogna:

- preparare una istruzione SQL *SELECT*;
- eseguirla con la funzione *mysql_query()*; nel caso di una istruzione *SELECT* la funzione *mysql_query()* restituisce un identificatore da utilizzare per accedere ai dati restituiti come risultato della query;
- accedere ai dati; le funzioni di accesso ai dati restituiscono un record ogni volta che sono richiamate e *false* quando non ci sono più dati; si possono usare le funzioni ***mysql_fetch_row()*** che restituisce un array che contiene in ogni elemento un campo del record o ***mysql_fetch_array()*** che restituisce un array associativo in cui le chiavi sono i nomi dei campi del record.

Esempio 10.6

Visualizzazione dei dati (nome città e CAP corrispondente) della tabella *elenco* del database.

```
<html>
<head><title>PHP</title></head>
<body>
<h1>Visualizzazione dati</h1>
<?php
        $conn = mysql_connect("localhost", "root", "");
        if (!$conn) die('Impossibile connettersi: '.mysql_error());
        mysql_select_db("dati") or die ('Impossibile aprire il database: '.mysql_error());
?>
```

Loescher Editore - Vietata la vendita e la diffusione

```
<table border="1">
<tr><th>Citta'</th><th>CAP</th></tr>
<?php
        $risultato = mysql_query("SELECT * FROM elenco");
        while($riga = mysql_fetch_array($risultato)) {
                print("<tr>");
                print("<td>".$riga["citta"]."</td>");
                print("<td>".$riga["cap"]."</td>");
                print("</tr>");
        }
        mysql_close($conn);
?>
</table>
</body>
</html>
```

Figura 10.7 *Esecuzione dell'esempio*

Inserimento di dati

Per inserire dati in una tabella del database bisogna:

- preparare una istruzione SQL *INSERT*;
- eseguirla con la funzione *mysql_query()*.

Esempio 10.7

Inserimento di un record.

```
<html>
<head><title>PHP</title></head>
<body>
<h1>Inserimento dati</h1>
<table>
<form method="post" action="inserimento.php">
<tr><td>Citta'</td><td><input type="text" name="citta"></td></tr>
<tr><td>CAP:</td><td><input type="text" name="cap"></td></tr>
</table>
<br/>
<input type="submit" value="Inserisci">
</form>
</body>
</html>
```

Figura 10.8 *Esecuzione dell'esempio*

file inserimento.php

```
<html>
<head><title>PHP</title></head>
<body>
<h1>Inserimento dati</h1>
<?php
        $conn = mysql_connect("localhost", "root", "");
        if (!$conn) die('Impossibile connettersi: '.mysql_error());
        mysql_select_db("dati") or die ('Impossibile aprire il database: '.mysql_error());
```

Loescher Editore - Vietata la vendita e la diffusione

```
        if(trim($_POST['citta']) == "") {
?>
            <p>La citta' deve essere specificata</p>
            <a href="inserimento.html">Ripeti</a>
<?php
}
        else {
            $insert = "INSERT INTO elenco (citta, cap) VALUES ('$citta','$cap')";
            if (mysql_query($insert))
                print("Inserimento riuscito");
            else
                print("Inserimento non riuscito");
        }
mysql_close($conn);
?>
</body>
</html>
```

Modifica di dati

Per modificare i dati di una tabella del database bisogna:

- preparare una istruzione SQL *UPDATE*;
- eseguirla con la funzione *mysql_query()*.

La funzione *mysql_affected_rows()* restituisce il numero di righe effettivamente aggiornate.

Gestione database via ODBC

PHP permette di gestire un database via ODBC con funzioni analoghe a quelle viste per MySQL.

odbc_connect*(DSN, utente, password)*	apre una connessione con il database specificato mediante l'origine dati (*DSN*) e restituisce un identificativo di connessione;
odbc_exec*(connessione, query)*	esegue la query (nel caso di una istruzione *SELECT* restituisce un identificatore da utilizzare per accedere ai dati);
odbc_fetch_row*(risultato)*	restituisce un array che contiene in ogni elemento un campo del record;
odbc_fetch_array*(risultato)*	restituisce un array associativo in cui le chiavi sono i nomi dei campi del record.

INSTALLAZIONE DI TOMCAT

Nella directory di installazione (per esempio *C:\Tomcat*) vengono create le directory:

bin	codice binario eseguibile e script;
common	classi disponibili sia alle classi interne che alle applicazioni Web;
conf	file di configurazione;
logs	file di log;
server	classi interne;

Loescher Editore - Vietata la vendita e la diffusione

shared	classi condivise da tutte le applicazioni Web;
classes	classi condivise non compresse;
lib	classi condivise in file JAR;
webapps	applicazioni Web;
work	file di lavoro e temporanei di *Tomcat*;
temp	file temporanei della JVM.

I file delle applicazioni vanno inseriti nella directory principale del server Web: *C:\Tomcat\webapps*.

Struttura delle applicazioni Web

Ogni applicazione è costituita da una directory principale contenente:

file ***.jsp***
file ***.html***
file ***.gif*** e ***.jpeg***
altri file che vengono inviati al browser;
WEB-INF/ file non visibili al browser;
- ***web.xml*** file di configurazione dell'applicazione;
- ***classes/*** file *.class* dell'applicazione.

10.6 JSP E SERVLET

Servlet e JSP sono tecnologie che utilizzano **Java** per la realizzazione di **applicazioni server side**. L'uso di servlet e JSP come tecnologie server side ha molti vantaggi che derivano dall'uso di Java: si tratta di tecnologie freeware e multipiattaforma, portabili, orientate agli oggetti, che permettono di riutilizzare il codice (grazie anche all'uso dei JavaBeans).

Un **servlet** è un'applicazione Java in esecuzione su una JVM (Java Virtual Machine) residente su un server.

I **servlet** sono classi Java che vengono compilate per produrre file *.class* eseguiti sul server; l'uso dei servlet richiede una buona conoscenza della programmazione Java.

La tecnologia **JSP** (JavaServer Pages) rappresenta una evoluzione dei servlet.
Le pagine JSP possono fare le stesse cose dei servlet ma sono più semplici da realizzare.
Le pagine JSP vengono tradotte automaticamente in servlet dal server Web.

Al momento della richiesta di una pagina JSP viene eseguito un servlet. Ogni pagina in realtà diventa un servlet. La differenza è nel modo in cui il servlet viene creato; non si deve programmare il servlet direttamente ma la creazione del servlet è automatica.

Nel server Web esiste un motore JSP (**JSPContainer**) che esegue le operazioni di **page translation** (creazione del sorgente Java del servlet corrispondente alla pagina JSP) e **page compilation** (creazione del file *.class* del servlet).

Queste operazioni avvengono solo alla prima richiesta della pagina; alla prima richiesta il tempo di risposta è maggiore perché il servlet deve essere compilato.

Nota

Loescher Editore - Vietata la vendita e la diffusione

Richiesta di pagine JSP

La prima volta che si effettua la richiesta di una pagina JSP viene **creato e avviato il servlet** corrispondente. Il servlet rimane in memoria per servire le richieste successive.
Se la pagina non è stata modificata, le richieste successive richiamano il servlet già compilato.
Se la pagina è stata modificata, alla prima richiesta viene creato il nuovo servlet.

Il codice prodotto a partire dalla pagina JSP può contenere degli **errori di compilazione**.
In tal caso alla richiesta della pagina il server risponde con una pagina "*Server Error*" (errore 500) con il dettaglio degli errori di compilazione.

La prima pagina JSP

Esempio 10.8

Generazione di una pagina Web.

```
<html>
<head><title>JSP</title></head>
<body>
<% out.println("<h1>JSP</h1>"); %>
<% out.println("Prima pagina"); %>
</body>
</html>
```

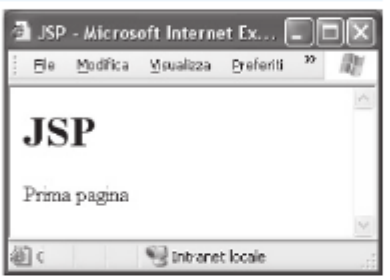


Figura 10.9 *Esecuzione dell'esempio*

out è un oggetto predefinito che permette di generare il codice HTML della pagina di risposta con i metodi *print()* e *println()*.

Il codice HTML può essere inserito direttamente nella pagina o può essere stampato sull'oggetto *out* con i metodi *print()*.

Un esempio di servlet generato dal JSPContainer a partire da una pagina JSP

Per una pagina che si chiama *pagina.jsp* viene creato il servlet *pagina_jsp.java*, poi compilato in *pagina_jsp.class*.

Nota Se la pagina si trova in *webapps\JSP* e Tomcat è installato in *C:\Tomcat* il servlet viene creato in *C:\Tomcat\work\Catalina\localhost\JSP\org\apache\jsp*.

La pagina *prima.jsp* genera il servlet *prima_jsp.java*:

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class prima_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {

    private static java.util.List _jspx_dependants;

    public Object getDependants() {
        return _jspx_dependants;
    }
```

Loescher Editore - Vietata la vendita e la diffusione

```
    public void _jspService(HttpServletRequest request, HttpServletResponse response)
        throws java.io.IOException, ServletException {

        JspFactory _jspxFactory = null;
        PageContext pageContext = null;
        HttpSession session = null;
        ServletContext application = null;
        ServletConfig config = null;
        JspWriter out = null;
        Object page = this;
        JspWriter _jspx_out = null;
        PageContext _jspx_page_context = null;

        try {
            _jspxFactory = JspFactory.getDefaultFactory();
            response.setContentType("text/html");
            pageContext = _jspxFactory.getPageContext(this, request, response,
                null, true, 8192, true);
            _jspx_page_context = pageContext;
            application = pageContext.getServletContext();
            config = pageContext.getServletConfig();
            session = pageContext.getSession();
            out = pageContext.getOut();
            _jspx_out = out;

            out.write("<html>\n");
            out.write("<head><title>JSP</title></head>\n");
            out.write("<body>\n");
            out.println("<h1>JSP</h1>");
            out.write('\n');
            out.println("Prima pagina");
            out.write("\n");
            out.write("</body>\n");
            out.write("</html>\n");
        } catch (Throwable t) {
            if (!(t instanceof SkipPageException)){
                out = _jspx_out;
                if (out != null && out.getBufferSize() != 0)
                    out.clearBuffer();
                if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
            }
        } finally {
            if (_jspxFactory != null) _jspxFactory.releasePageContext(_jspx_page_context);
        }
    }
}
```

Il metodo ***_jspService()*** contiene il codice della pagina JSP e fornisce l'accesso a tutti gli oggetti predefiniti utilizzabili nella pagina.
Gli oggetti ***request*** e ***response*** sono parametri di *_jspService()*.
L'oggetto ***page*** è impostato a *this*.
Il metodo *getPageContext()* di *JspFactory* permette di ottenere un oggetto ***PageContext*** che poi viene utilizzato per ottenere gli oggetti ***application***, ***config***, ***session*** e ***out***.

Loescher Editore - Vietata la vendita e la diffusione

Primi elementi

I tag Java in una pagina JSP comprendono scriptlet, espressioni, dichiarazioni, direttive e azioni.

Commenti

In una pagina JSP si possono inserire diversi tipi di **commenti**:

commenti HTML	compresi tra i tag <!-- e -->, che compaiono nel codice HTML generato;
commenti JSP	relativi al codice della pagina JSP, compresi tra i tag <%-- e --%>, che non vengono inseriti nel codice HTML generato;
commenti Java	usati per commentare il codice Java inserito nella pagina; i commenti di una sola riga iniziano con //, i commenti di più righe sono compresi tra /* e */; durante la page translation vengono inseriti nel codice del servlet.

Scriptlet

<% *scriptlet* %>

Gli **scriptlet** permettono di inserire nelle pagine codice Java. Negli scriptlet si possono usare strutture di controllo e dichiarazioni di variabili (ma non dichiarazioni di metodi).

Espressioni

<%= *espressione* %>

Le **espressioni** vengono valutate, convertite in *String* e stampate sull'oggetto *out*.

```
<%= "La data corrente è:" %>
<%= new java.util.Date() %>
```

Dichiarazioni

<%! *dichiarazione* %>

Le **dichiarazioni** permettono di definire variabili e metodi.

Dichiarazione di un array con i nomi dei mesi:

```
<%! String[] mesi = {"Gennaio","Febbraio","Marzo","Aprile","Maggio","Giugno",
"Luglio","Agosto","Settembre","Ottobre","Novembre","Dicembre"}; %>
```

Direttive

<%@ *direttiva attributi* %>

Le **direttive** sono indicazioni di carattere generale.

La direttiva ***page*** permette di specificare impostazioni che influiscono sulla compilazione della pagina. L'attributo *import* specifica quali package devono essere importati.

La direttiva ***include*** permette di importare una pagina in un'altra; l'inclusione avviene in fase di page translation. Si possono includere altre pagine JSP ma anche file HTML o di testo.
La direttiva *include* è utile per importare in pagine diverse codice comune.

Azioni

<jsp:*azione parametri*>

Le **azioni** permettono di ampliare le funzionalità delle pagine JSP.

Loescher Editore - Vietata la vendita e la diffusione

L'azione ***jsp:include*** permette di includere in una pagina l'output di un'altra pagina JSP.

La differenza tra l'azione *include* e la direttiva *include* è che l'azione *include* esegue la pagina specificata e ne include l'output (include un documento dinamico, quindi può essere solo una pagina JSP) mentre la direttiva *include* include il testo della pagina (un documento statico, che quindi può essere una pagina JSP ma anche codice HTML). **Nota**

L'azione ***jsp:forward*** consente di inoltrare la richiesta a un'altra risorsa (pagina JSP, ma anche HTML).

L'azione ***jsp:plugin*** permette di inserire un'applet in una pagina JSP.

L'azione ***jsp:useBean*** permette di usare un bean in una pagina; un bean è una classe Java.

L'uso dei bean permette di suddividere nelle applicazioni il livello di presentazione dal livello della logica di elaborazione. **Nota**

Tecnologia JavaBeans

La tecnologia JavaBeans è uno standard per la scrittura di classi Java.
Perché una classe Java sia un bean nel senso della tecnologia JavaBeans la classe deve:

- avere un costruttore senza parametri;
- avere per ogni *attributo* un metodo *setAttributo()* e un metodo *getAttributo()*.

Oggetti predefiniti

Le pagine JSP possono usare vari oggetti predefiniti (o impliciti):

request	rappresenta la richiesta inviata dal browser;
response	rappresenta la risposta da restituire al browser;
session	rappresenta una sessione;
application	rappresenta l'applicazione;
out	rappresenta lo standard output;
page	rappresenta la pagina stessa;
pageContext	rappresenta il contesto di esecuzione della pagina;
config	rappresenta un oggetto di configurazione della pagina.

Request

L'oggetto predefinito ***request*** permette di accedere alle informazioni sulla richiesta inviata dal browser.

Parametri della richiesta

String ***getParameter****(String name)*	restituisce il valore del parametro specificato (o *null* se il parametro non esiste);
Enumeration ***getParameterNames****()*	restituisce una *Enumeration* di stringhe con i nomi dei parametri.

Questi metodi permettono di accedere ai parametri sia che siano passati da un URL che da un modulo, sia col metodo *GET* che col metodo *POST*.

L'oggetto *request* permette anche di ottenere informazioni sui cookie e sulle sessioni. **Nota**

Loescher Editore - Vietata la vendita e la diffusione

Esempio 10.9

Pagina che stampa i valori dei parametri passati con la richiesta.

```
<%@ page import="java.util.Enumeration" %>
<html>
<head><title>JSP</title></head>
<body>
<h1>Parametri della richiesta</h1>
<%
Enumeration parametri = request.getParameterNames();
while (parametri.hasMoreElements()) {
        String nome = (String)parametri.nextElement();
        String valore = request.getParameter(nome);
%>
<%= nome + ": " %>
<%= valore + "<br/>" %>
</tr>
<% } %>
</body>
</html>
```

Figura 10.10
Esecuzione dell'esempio con i dati passati dal modulo

Modulo di richiesta

```
<html>
<head><title>JSP</title></head>
<body>
<h1>Modulo</h1>
<form method="post" action="parametriRequest.jsp">
<table border="1">
<tr><td>Cognome:</td>
<td><input type="text" name="cognome"><br></td></tr>
<tr><td>Nome:</td>
<td><input type="text" name="nome"><br></td></tr>
</table><br/>
<input type="submit" value="Invia">
/form
</body>
</html>
```

Pagina che crea un modulo e lo elabora

Può essere utile riunire nella stessa pagina JSP il codice che crea un modulo e il codice che lo elabora.
Per vedere se l'utente ha inviato il modulo basta dare un nome al pulsante di invio e controllare se il valore corrispondente nel parametro della richiesta è diverso da *null*.

```
if request.getParameter("pulsanteInvio") != null {
        //elaborazione del modulo
}
```

Response

L'oggetto predefinito ***response*** fornisce tutti gli strumenti per creare e inviare una risposta al browser.

Loescher Editore - Vietata la vendita e la diffusione

Nota

La risposta è costituita dai risultati dell'esecuzione della pagina JSP.

L'oggetto *response* non viene utilizzato per la composizione della pagina. A questo scopo si usano le espressioni, si scrive l'output sull'oggetto *out* o si scrivono direttamente i tag HTML.

L'oggetto *response* permette di impostare le **intestazioni HTTP** che servono per l'invio della pagina.

Nota

L'oggetto *response* permette anche di impostare cookie e di codificare le informazioni di sessione negli URL.

Esempio 10.10

Pagina che rinvia ad una nuova pagina dopo 5 secondi.

```
<html>
<head><title>JSP</title></head>
<body>
Attendere ...
<% response.setHeader("Refresh", "5; URL=pagina.html");%>
</body>
</html>
```

Figura 10.11 *Esecuzione dell'esempio*

Cookie

Per **inviare un cookie** al browser si usa il metodo *addCookie()* dell'oggetto *response*.
Un cookie è un oggetto della classe ***Cookie***.
Per **leggere i cookie** in una pagina si usa il metodo *getCookies()* dell'oggetto *request*.

Esempio 10.11

Uso dei cookie.

Pagina per la memorizzazione di alcuni cookie per la durata della sessione

```
<html>
<head><title>JSP</title></head>
<body>
<h1>Cookie: memorizzazione</h1>
<%
response.addCookie(new Cookie("Colore","rosso"));
response.addCookie(new Cookie("Fiore","garofano"));
%>
</body>
</html>
```

Pagina per la visualizzazione del nome e del valore di tutti i cookie

```
<html>
<head><title>JSP</title></head>
<body>
<h1>Cookie: visualizzazione</h1>
<%
Cookie[] cookies = request.getCookies();
if (cookies == null) {
        out.print("Non ci sono cookie");
}
```

Figura 10.12 *Esecuzione dell'esempio*

Loescher Editore - Vietata la vendita e la diffusione

```
else {
        for(int i=0; i<cookies.length; i++) {
                out.print(cookies[i].getName() + ": ");
                out.print(cookies[i].getValue() + "<br/>");
        }
}
%>
</body>
</html>
```

Sessioni

L'oggetto predefinito ***session*** permette di gestire le sessioni.

L'oggetto *session* sul server è unico per ciascun utente ed è disponibile solo se l'attributo ***session*** della direttiva ***page*** ha il valore *true* (il valore è *true* per default).

Si possono ottenere l'oggetto *session*, l'ID di sessione o informazioni sulla sessione con i metodi dell'oggetto *request*.

Si possono **gestire le sessioni** e memorizzare **attributi di sessione** (cioè valori validi per la durata della sessione) con i metodi dell'oggetto *session*.

Esempio 10.12

Uso degli attributi di sessione.

Pagina per la memorizzazione di alcuni attributi di sessione

```
<html>
<head><title>JSP</title></head>
<body>
<h1>Sessioni: memorizzazione valori</h1>
<%
String AttrNome1 = "Utente";
String AttrValore1 = "Rossi";
String AttrNome2 = "Password";
String AttrValore2 = "S&gr&t0";
session.setAttribute(AttrNome1,AttrValore1);
session.setAttribute(AttrNome2,AttrValore2);
%>
</body>
</html>
```

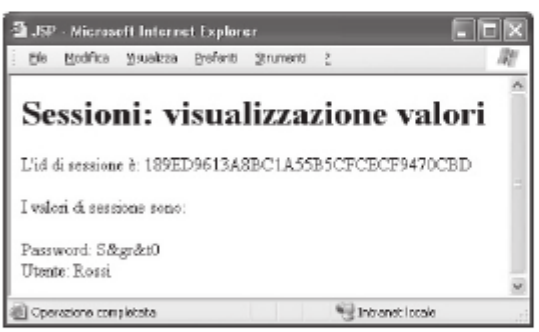


Figura 10.13 *Esecuzione dell'esempio*

Pagina per la visualizzazione del nome e del valore di tutti gli attributi di sessione

```
<%@ page import="java.util.Enumeration" %>
<html>
<head><title>JSP</title></head>
<body>
<h1>Sessioni: visualizzazione valori</h1>
<p>L'id di sessione è: <%=  request.getRequestedSessionId() %></p>
<p>I valori di sessione sono:</p>
<%
Enumeration valoriSessione = session.getAttributeNames();
```

Loescher Editore - Vietata la vendita e la diffusione

```
while (valoriSessione.hasMoreElements()) {
        String nome = (String)valoriSessione.nextElement();
        String valore = (String)session.getAttribute(nome);
%>
<%= nome + ": " %>
<%= valore + "<br/>" %>
<% } %>
</body>
</html>
```

Gestione degli errori nelle pagine JSP

Quando si verifica un errore di esecuzione il sistema genera una eccezione.

Se le eccezioni non vengono gestite, si propagano al servlet container e vengono riportate sul monitor del computer dove il server è attivo.

Nota Nelle pagine JSP, a differenza di Java, non è indispensabile gestire le eccezioni.

Perché gli errori non compaiano sul server ma possano essere segnalati all'utente con pagine apposite, bisogna gestire le eccezioni; si possono intercettare le eccezioni con istruzioni *try catch* nelle singole pagine o, meglio, usare pagine di errore.

Una **pagina di errore** è una pagina che contiene la direttiva ***page*** con l'attributo ***isErrorPage*** a *true*.

```
<%@ page isErrorPage = "true" %>
```

Una pagina fa **riferimento** a una pagina di errore con la direttiva ***page*** con l'attributo ***errorPage***.

```
<% page errorPage = "paginaErrore.jsp" %>
```

Quando le istruzioni della pagina che contiene questa direttiva sollevano un'eccezione, automaticamente viene richiamata la pagina di errore.

Nella pagina di errore è disponibile l'oggetto predefinito ***exception*** che rappresenta l'ultima eccezione sollevata in una pagina in cui il file è stato specificato come pagina di errore.

Nota L'oggetto *exception* è accessibile solo nelle pagine di errore (quelle in cui l'attributo *isErrorPage* della direttiva *page* è impostato a *true*).

Tramite l'oggetto *exception* si possono usare i metodi:

String ***getMessage()***	restituisce la descrizione associata all'eccezione al momento della creazione dell'oggetto;
void ***printStackTrace****(PrintWriter s)*	stampa lo stato dello stack sull'oggetto *Writer* specificato.

Esempio 10.13

Pagina di errore che stampa il messaggio di errore e la situazione dello stack.

```
<%@ page isErrorPage="true" %>
<%@ page import="java.io.PrintWriter" %>
<% PrintWriter writer = new PrintWriter(out); %>
```

Loescher Editore - Vietata la vendita e la diffusione

```
<html>
<head><title>JSP</title></head>
<body>
<h1> Errore nell'esecuzione della pagina</h1>
<p>Si è verificato l'errore: <b><%= exception.getMessage() %></b></p>
<p>Ecco lo stack:</p>
<% exception.printStackTrace(writer); %>
</body>
</html>
```

Anche il programmatore può generare un'eccezione per segnalare un errore o una situazione anomala.

Le pagine di errore possono essere utilizzate nella progettazione dell'applicazione per gestire situazioni di vario tipo, per esempio come risposta agli errori di input dell'utente.

10.7 GESTIONE DI DATABASE CON JDBC

Servlet e pagine JSP, come le applicazioni Java, usano l'**API JDBC** per accedere ai database.

Nota

Un servlet o una pagina JSP che deve usare l'API JDBC deve importare il package *java.sql*.

Eccezioni SQLException

Nelle pagine JSP non è necessario intercettare le eccezioni ***SQLException*** perché la pagina funzioni; conviene creare una **pagina di errore** per gestire tutte le eccezioni che si possono verificare.

Esempio 10.14

Visualizzazione dei dati (nome città e CAP corrispondente) della tabella *elenco* del database con DSN *origine*.

Figura 10.14 *Esecuzione dell'esempio*

file visualizzazione.jsp

```
<<%@page errorPage="../errore.jsp" %>
<%@ page language="java" import="java.sql.*" %>
<html>
<head><title>JSP</title></head>
<body>
<h1>Visualizzazione dati</h1>
<table border="1">
<tr><th>Città</th><th>CAP</th></tr>
<%
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:origine");
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("SELECT * FROM elenco");
while (rs.next()) {
    out.println("<tr>");
```

Loescher Editore - Vietata la vendita e la diffusione

```
        out.println("<td>" + rs.getString("citta") + "</td>");
        out.println("<td>" + rs.getString("cap") + "</td>");
        out.println("</tr>");
}
rs.close();
st.close();
con.close();
%>
</table>
</body>
</html>
```

Esempio 10.15

Inserimento di un record.

```
<html>
<head><title>JSP</title></head>
<body>
<h1>Inserimento dati</h1>
<table>
<form method="post" action="inserimento.jsp">
<tr><td>Città:</td><td><input type="text" name="città"></td></tr>
<tr><td>CAP:</td><td><input type="text" name="cap"></td></tr>
</table>
<br/>
<input type="submit" value="Inserisci">
</form>
</body>
</html>
```

Figura 10.15 *Esecuzione dell'esempio*

file inserimento.jsp

```
<%@page errorPage="../errore.jsp" %>
<%@ page language="java" import= "java.sql.*" %>
<html>
<head><title>JSP</title></head>
<body>
<h1>Inserimento dati</h1>
<%
String città = request.getParameter("città");
String cap = request.getParameter("cap");
if (città.equals("")) {
%>
        <p>La città deve essere specificata</p>
        <a href="inserimento.html">Ripeti</a>
<%
}
else {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection con = DriverManager.getConnection("jdbc:odbc:origine");
        PreparedStatement pst = con.prepareStatement("INSERT INTO elenco (citta, cap) VALUES(?, ?)");
        pst.setString(1, città);
```

Loescher Editore - Vietata la vendita e la diffusione

```
        pst.setString(2, cap);
        int esito = pst.executeUpdate();
        if (esito > 0) {
                out.print("Inserimento riuscito");
        }
        else {
                out.print("Inserimento non riuscito");
        }
        pst.close();
        con.close();
}
%>
</body>
</html>
```

Le istruzioni:

```
PreparedStatement pst = con.prepareStatement("INSERT INTO elenco (citta, cap) VALUES(?, ?)");
pst.setString(1, città);
pst.setString(2, cap);
int esito = pst.executeUpdate();
```

si possono anche scrivere come:

```
Statement st = con.createStatement();
String query = "INSERT INTO elenco (citta, cap) VALUES('" + città + "','" + cap + "')";
int esito = st.executeUpdate(query);
```

CONFIGURAZIONE DEL SERVER IIS (INTERNET INFORMATION SERVER)

Internet Information Server comprende i server Web, FTP, NNTP e SMTP (ma solo per l'invio di posta; non si possono creare caselle di posta per gli utenti).
Si può installare *IIS* da *Installazione applicazioni*, *Installazione componenti di Windows*; basta selezionare *IIS* e poi i componenti desiderati.
Si possono **configurare** e **gestire i server** da *Programmi*, *Strumenti di amministrazione*, *Gestione servizi Internet*. A sinistra, facendo doppio clic sul nome del server, si vedono i servizi gestiti; facendo clic destro su un servizio e scegliendo *Proprietà* si può configurare il servizio.
IIS viene caricato come **servizio** ed eseguito in **background**; dopo l'installazione tutti i servizi specificati sono attivi e in esecuzione.

Durante l'**installazione** del server Web si devono inserire l'indirizzo IP del server, il nome del computer, l'indirizzo di posta dell'amministratore del server Web e l'indirizzo IP del server DNS.
Per **provare** il funzionamento del server basta inserire nella barra indirizzi del browser l'indirizzo del computer su cui è attivo il server (in una intranet basta *http://nomecomputer*).

IIS memorizza una home page di default (*default.htm*) nella cartella *\inetpub\wwwroot*.
Tutte le pagine Web che devono essere pubblicate vanno inserite nella cartella *\inetpub\wwwroot*, eventualmente in sottocartelle. Alla richiesta dell'URL *http:/unadir/unfile.htm* viene cercato il file *c:\inetpub\wwwroot\unadir\unfile.htm*.
Viene creata automaticamente anche una cartella *scripts* con le autorizzazioni di *Esecuzione* (non di *Lettura*) per gli script.

Loescher Editore - Vietata la vendita e la diffusione

Il server Web di *IIS* permette l'esecuzione di applicazioni ASP.

Si può **configurare** il server Web da *Programmi*, *Strumenti di amministrazione*, *Gestione servizi Internet*, facendo clic destro sul server Web e scegliendo *Proprietà*.
La finestra *Proprietà* presenta varie schede.

La **scheda *Sito Web*** contiene:

- l'identificazione del server (descrizione, indirizzo IP e porta);
- la sezione *Connessioni* che permette di scegliere tra connessioni *Illimitate* o *Limitate a* (indicando il numero) e di indicare il *Timeout*, cioè il tempo di inattività dopo cui ci si disconnette;
- la sezione *Consenti registrazione attività* che attiva la scrittura in file di log per sapere chi accede al server (da quali indirizzi IP) e quando; abilitando i registri del server si ottengono molte informazioni sulle richieste degli utenti (si vedono le pagine in cui gli utenti hanno fatto clic per arrivare al sito, le stringhe di ricerca usate nei motori di ricerca, e quale motore hanno usato, le aree visitate ecc.); è quindi possibile analizzare le richieste e migliorare il sito di conseguenza.

La **scheda *Home directory*** definisce la **directory radice** del server Web che per default è *c:\inetpub\wwwroot*; presenta inoltre quattro caselle di selezione che permettono di abilitare i **tipi di accesso**:

accesso origine script
si può vedere il sorgente degli script (anche quello delle pagine ASP);
lettura si possono vedere i file ma non si possono fare modifiche o aggiungere altri file;
scrittura si possono aggiungere altri file (cioè effettuare upload) o modificare i file;
esplorazione directory
se manca il file di default, si ottiene un elenco del contenuto della directory.

La **scheda *Documenti*** permette di abilitare l'uso di pagine predefinite con la casella di selezione *Abilita documenti predefiniti* e di specificare i nomi dei file di default (di solito *default.htm* o *index.htm* ma anche quelli predefiniti di ASP: *default.asp* e *iisstart.asp*).

La **scheda *Errori personalizzati*** permette di specificare il file HTML da restituire per ogni specifico errore HTTP.

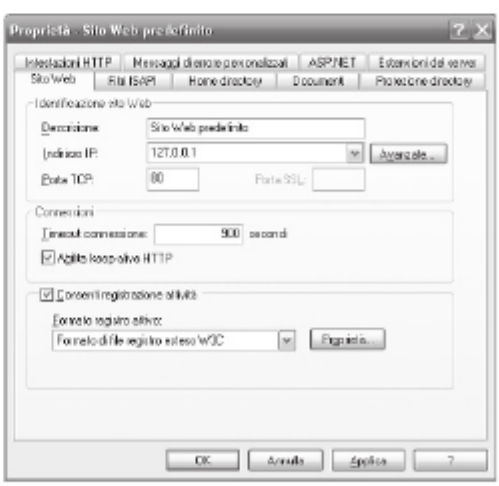


Figura 10.16 *La scheda Sito Web della finestra delle proprietà di un server Web*

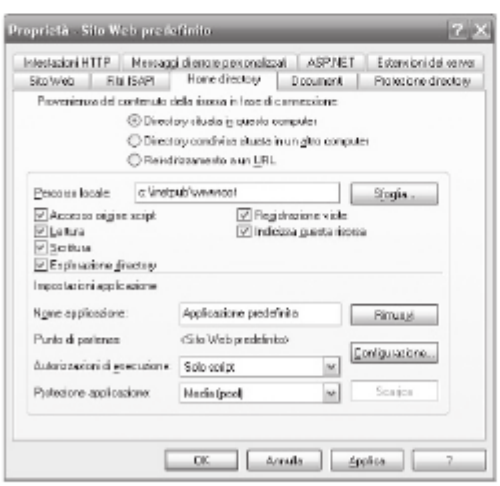


Figura 10.17 *La scheda Home directory della finestra delle proprietà di un server Web*

Loescher Editore - Vietata la vendita e la diffusione

10.8 ASP

Il codice va inserito in una pagina HTML racchiuso tra i tag ***<%*** e ***%>***.
Le pagine hanno estensione ***.asp***.

Il linguaggio di script predefinito è ***VBScript***; si può specificare un linguaggio diverso (come *JavaScript*) all'inizio della pagina con la direttiva:

```
<% LANGUAGE="linguaggio" %>
```

o con il tag *<script>*:

```
<script language="linguaggio" runat="server">...</script>
```

Si possono usare tutte le istruzioni del linguaggio di script.

Le variabili usate in una pagina hanno come campo di validità la pagina stessa.

Oggetti predefiniti

Le pagine ASP possono usare vari oggetti predefiniti:

Request	rappresenta la richiesta inviata dal browser;
Response	rappresenta la risposta da restituire al browser;
Session	rappresenta una sessione;
Application	rappresenta l'applicazione;
Server	rappresenta l'ambiente di esecuzione ASP.

Questi oggetti hanno metodi, proprietà e collezioni che si utilizzano con la notazione puntata:

```
oggetto.proprietà
oggetto.metodo()
oggetto.collezione
```

Nota

Una collezione è un oggetto che rappresenta o contiene un gruppo di oggetti simili; gli oggetti contenuti nella collezione sono identificati mediante un indice numerico o alfanumerico (array associativi).

Response

L'oggetto ***Response*** fornisce tutti gli strumenti per creare e inviare una risposta al browser; la risposta è costituita dai risultati dell'esecuzione della pagina ASP.

Il metodo *Response.**Write**()* scrive le informazioni sulla pagina Web da inviare al browser.

```
<% Response.Write "<H1>titolo</H1>" %>
```

Una abbreviazione di *Response.write(espressione)* è ***<%= espressione %>***.

```
<%= variabile>
```

manda al browser il valore della variabile.

Esempio 10.16

Generazione di una pagina Web.

```
<html>
<head><title>ASP</title></head>
```

Loescher Editore - Vietata la vendita e la diffusione

```
<body>
<%
        Response.Write("<h1>ASP</h1>")
        Response.Write("Prima pagina")
%>
</body>
</html>
```

Figura 10.18 *Esecuzione dell'esempio*

Request

L'oggetto predefinito ***Request*** permette di accedere alle informazioni sulla richiesta inviata dal browser.

La collezione *Request.**QueryString*** contiene i parametri di una richiesta che usa il metodo *GET*. La collezione *Request.**Form*** contiene i valori inseriti nel modulo usato per una richiesta che usa il metodo *POST*.

```
<%= Request.QueryString(nome) %>
<%= Request.Form(nome) %>
```

Indipendentemente dal metodo (*GET* o *POST*) usato per la richiesta si può fare riferimento a un elemento di un modulo con:

```
Request(nome)
```

Cookie

Per gestire i cookie si usa la collezione ***Cookies***; si può impostare un cookie usando *Response.Cookies*. Si può impostare una data di scadenza utilizzando la proprietà *expires* del cookie.
Il cookie deve essere la prima cosa inviata al browser.

```
<% Response.Cookies("colore")="rosso"
Response.Cookies("colore").expires =#December 31, 2008 00:00:00#%>
```

Si può leggere un cookie usando *Request.Cookies*.

```
<%=  Request.Cookies("colore") %>
```

Sessioni

Per gestire la sessione viene memorizzato un cookie sul computer dell'utente che contiene l'ID di sessione.
L'oggetto predefinito ***Session*** permette di memorizzare e poi riutilizzare le variabili di sessione.

```
<% Session("utente")="Rossi" %>

<%= Session("utente") %>
```

10.9 GESTIONE DI DATABASE IN ASP CON ADO

ASP si serve della tecnologia **ADO** (ActiveX Data Object) per utilizzare i driver ODBC per accedere al database.

Loescher Editore - Vietata la vendita e la diffusione

Gli oggetti principali del modello ADO sono:

Connection	permette di stabilire la connessione all'origine dati;
RecordSet	rappresenta un insieme di record di una tabella.

Si possono creare gli oggetti ADO con il metodo *Server.CreateObject()*.

Per prima cosa bisogna aprire una connessione; si può creare un oggetto ***Connection*** e usare il suo metodo ***Open()***.

```
Set conn = server.CreateObject("ADODB.Connection")
conn.Open "FILEDSN=origine.dsn"
```

Il metodo *Open()* permette di specificare una stringa che contiene informazioni di connessione, un nome utente e una password per accedere al database.

```
Open ConnectionString, UserId, Password
```

Per **specificare il database** si può usare:

- una origine dati ODBC (DSN):

  ```
  conn.Open "DSN=nome.dsn"
  ```

  o semplicemente:

  ```
  conn.Open("nome.dsn")
  ```

- una stringa di connessione dettagliata:

  ```
  conn.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" & Server.MapPath("nome.mdb")
  ```

Al termine bisogna chiudere la connessione usando il metodo *Close()* dell'oggetto *Connection* o impostando la variabile che rappresenta la connessione a *nothing*.

```
conn.Close
```

o

```
Set conn = Nothing
```

Per realizzare un'operazione sul database si può creare un oggetto ***RecordSet*** e usare il suo metodo ***Open()*** per eseguire un comando **SQL**.

```
Set recordset = Server.CreateObject("ADODB.Recordset")
query = stringa SQL
recordset.Open query, conn
```

Al termine si può chiudere il recordset usando il metodo *Close()* dell'oggetto *RecordSet* o impostando la variabile che rappresenta il recordset a *nothing*.

```
recordset.Close
```

o

```
Set recordset = Nothing
```

Si può ottenere un oggetto *RecordSet* anche eseguendo un comando SQL con il metodo ***Execute()*** di *Connection*.

```
query = stringa SQL
Set recordset = conn.Execute(query)
```

Loescher Editore - Vietata la vendita e la diffusione

Dopo aver creato il *RecordSet* si possono visualizzare i record scorrendoli mediante il metodo ***MoveNext()*** dell'oggetto *RecordSet*.
Per verificare se si è arrivati all'ultimo record si può testare la proprietà ***EOF*** dell'oggetto *RecordSet*.

Esempio 10.17

Visualizzazione dei dati (nome città e CAP corrispondente) della tabella *elenco* del database.

```
<html>
<head>
<title>ASP</title>
</head>
<body>
<%
        set conn = Server.CreateObject("ADODB.Connection")
        conn.Open "DSN=origine.dsn"
        set recordset = Server.CreateObject("ADODB.Recordset")
%>
<table border="1">
<tr><th>Citta'</th><th>CAP</th></tr>
<%
        query = "SELECT * FROM elenco"
        recordset.Open query, conn
        do while not recordset.EOF
%>
                <tr>
                <td><%= recordset("citta") %></td>
                <td><%= recordset("cap") %></td>
                </tr>
<%
                recordset.MoveNext
        Loop
%>
</table>
</body>
</html>
```

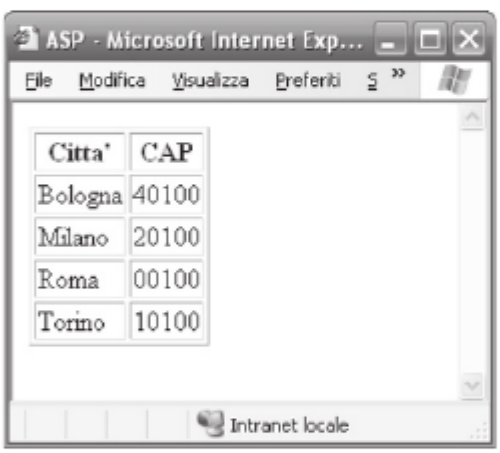


Figura 10.19 *Esecuzione dell'esempio*

Creazione di record

Per creare nuovi record si può usare il metodo ***AddNew()*** dell'oggetto *RecordSet*.

```
recordset.AddNew
recordset("citta") = "Padova"
recordset("cap") = "35100"
recordset.Update
```

AddNew crea un nuovo record;
Update trasferisce il contenuto del record al database e lo memorizza.

In alternativa si può eseguire un comando SQL *INSERT* usando il metodo *Execute()* di *Connection*.

Loescher Editore - Vietata la vendita e la diffusione

Modifica di record

Per modificare un record basta modificarne i campi desiderati e aggiornare il record con il metodo ***Update()*** di *RecordSet*.

```
recordset("cap") = "35138"
recordset.Update
```

In alternativa si può eseguire un comando SQL *UPDATE* usando il metodo *Execute()* di *Connection*.

Per poter effettuare operazioni di scrittura nel database (inserimenti o modifiche) bisogna specificare l'opzione *locktype* (il valore di default consente solo operazioni di lettura):

```
recordset.Open query, conn, , adLockOptimistic
```

10.10 ASP.NET

La piattaforma .NET

La piattaforma **.NET** è una piattaforma per creare ed eseguire applicazioni indipendentemente dal linguaggio di programmazione utilizzato (e teoricamente dal sistema operativo). I programmi scritti in qualsiasi linguaggio di programmazione (per ora quelli supportati sono Visual Basic, Visual C++ e C#) vengono compilati in un codice intermedio che viene poi eseguito dal **CLR** (Common Language Runtime), una macchina virtuale che si frappone fra l'applicazione e il sistema operativo).

.NET si basa sui **Web service** che permettono la comunicazione anche tra applicazioni che si trovano su computer diversi (elaborazione distribuita); la comunicazione tra le applicazioni avviene mediante il protocollo **SOAP** (Simple Object Application Protocol) che usa HTTP per il trasporto dei dati e XML per la descrizione dei dati da comunicare.

Il modello di programmazione di .NET è chiamato **framework .NET** e fornisce un'API che espande le API di *Windows* per fornire il supporto ai Web service.

.NET Framework 3.0 (inizialmente chiamato WinFx!) è disponibile per *Windows Vista*, *Windows Server 2003/2008* e *Windows XP*; le applicazioni possono essere realizzate rapidamente con Visual Studio 2005/2008.

Le pagine ASP.NET

Il suffisso .NET (si legge dot net) di ASP.NET si riferisce al **framework .NET**, una libreria di classi raggruppate in gruppi logici chiamati spazi dei nomi (**namespace**).

Nota

Il framework .NET deve essere installato sul computer che ospita il server Web per poter utilizzare le pagine ASP.NET.

In una applicazione si possono utilizzare contemporaneamente pagine ASP e ASP.NET.

Le pagine ASP.NET hanno estensione ***.aspx***.
Nelle pagine ASP.NET il codice di programmazione e il codice HTML sono divisi in due sezioni diverse, separando così la logica dalla presentazione.
Per il codice si possono utilizzare diversi linguaggi, in genere si utilizza **Visual Basic** o **C#**.
All'inizio di una pagina si può specificare il linguaggio utilizzato con la direttiva *page*.

Loescher Editore - Vietata la vendita e la diffusione

Il paradigma di programmazione usato è quello **a oggetti**; l'esecuzione è guidata dagli **eventi**.

ASP.NET inserisce automaticamente nelle pagine degli script lato client che inviano informazioni al server sugli eventi che si verificano e sullo stato dell'applicazione (risolvendo così anche il problema relativo al fatto che il protocollo HTTP è privo di stati).

Le pagine ASP.NET vengono tradotte in classi del linguaggio utilizzato e **compilate** alla prima richiesta. Una pagina viene ricompilata ogni volta che viene modificata.
Se il codice prodotto a partire dalla pagina contiene degli **errori di compilazione**, alla richiesta della pagina il server risponde con il dettaglio degli errori di compilazione.

Nota

Negli esempi del capitolo viene usato Visual Basic.
Il codice Visual Basic prodotto viene memorizzato in una cartella che dipende dalla directory di installazione di *Windows* e dalla versione del framework, per esempio *C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files\root*.

Configurazione

Il comportamento di una applicazione ASP.NET può essere configurato mediante i **file XML**:

machine.config	impostazioni con valori di default per il computer;
web.config	valori specifici per l'applicazione; può esserci un file nella directory root dell'applicazione e uno in ogni sottodirectory; le impostazioni sono la somma delle impostazioni applicate dai vari file che si incontrano in modo gerarchico dalla root alla directory della pagina.

Si può abilitare il **debug** e il **tracing** a livello di applicazione con le istruzioni:

```
<configuration>
    <system.web>
        <compilation debug="true" />
        <trace enabled="true" pageOutput="true" />
    </system.web>
</configuration>
```

pageOutput="true" fa in modo che le informazioni di trace siano visualizzate in fondo a ciascuna pagina, altrimenti ciascuna richiesta di pagina inoltra le informazioni di trace al trace viewer che può essere visualizzato richiamando nella barra indirizzi del browser *directoryApplicazione/**trace.axd***.

L'istruzione

```
<customErrors mode="RemoteOnly"
defaultRedirect="/messaggio.aspx" />
```

permette di inviare agli utenti remoti una pagina di errore personalizzata; gli utenti locali (in pratica gli sviluppatori) invece ricevono la pagina di default di ASP.NET con le informazioni sull'errore. Usando *mode="on"* la pagina di errore personalizzata viene visualizzata anche localmente; il default è *mode="off"* che visualizza la pagina di default anche per gli utenti remoti.

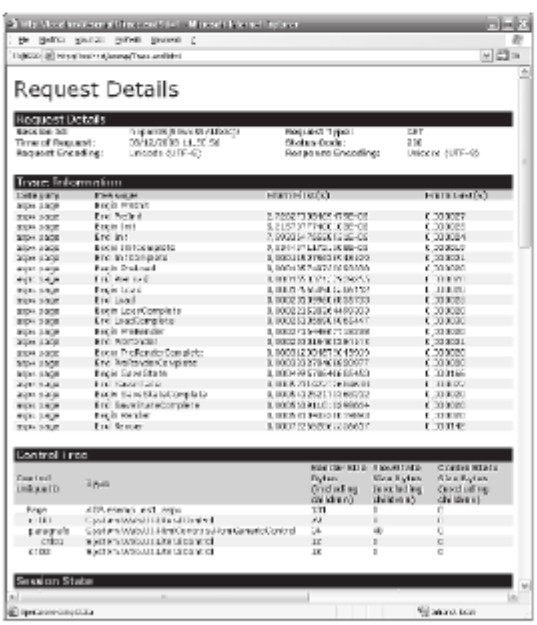


Figura 10.20
La pagina con le informazioni di trace visualizzate da trace.axd

Loescher Editore - Vietata la vendita e la diffusione

Nella directory dell'applicazione possono esserci alcune directory speciali: per esempio è possibile inserire i file di database e XML nella directory ***App_Data*** in modo che tali file non siano accessibili attraverso richieste HTTP al server.

I primi elementi

Una pagina è costituita da **tre sezioni**: direttive di pagina, sezione di codice e sezione di layout.

Direttive

Le **direttive** sono indicazioni di carattere generale.

La direttiva ***@Page*** permette di specificare impostazioni che influiscono sulla compilazione della pagina. L'attributo *language* specifica il linguaggio utilizzato; se non è specificato viene utilizzato di default Visual Basic.NET. L'attributo *Debug="True"* consente di avere maggiori informazioni di debug in caso di errori durante l'esecuzione della pagina. L'attributo *Trace="True"* consente di abilitare il trace per la pagina, ottenendo le informazioni in fondo alla pagina stessa.

La direttiva ***@Import*** con l'attributo *namespace* permette di importare un namespace in modo che a tutti i tipi definiti nel namespace si possa accedere senza specificare il nome completamente qualificato.

Codice

Il **codice** viene inserito in un tag ***<script>*** con l'attributo *runat="server"* ed è costituito dai gestori degli eventi.
Gli eventi sono relativi alle azioni eseguite sulla pagina, ma anche al ciclo di vita della pagina; in particolare l'evento ***Load*** viene generato al caricamento della pagina.

Layout

La sezione **layout** di pagina contiene codice HTML e **controlli**. I controlli sono oggetti lato server che vengono trasformati automaticamente in codice HTML (o meglio XHTML) e script lato client. Perché un elemento sia considerato lato server deve avere l'attributo ***runat="server"***.

I **controlli** sono di due tipi:

- controlli **HTML server**: tag HTML standard con in più l'attributo *runat="server"*;
- controlli **Web server**, identificati dal prefisso ***asp***: comprendono controlli corrispondenti ai tag HTML con funzionalità più estese, ma anche molti altri tipi di controlli come i controlli di validazione, i controlli data bound, i controlli data source e controlli vari come *AdRotator*, *Calendar* e *Xml*.

La gestione dei **form** avviene con la modalità **postback** cioè la pagina che contiene il form viene automaticamente rimandata a se stessa.

Controlli HTML server

I controlli **HTML server** sono tag HTML con in più l'attributo ***runat="server"*** che trasforma l'elemento HTML in un componente lato server, cioè un oggetto con proprietà corrispondenti agli attributi del tag HTML.

Esempio 10.18

Generazione di una pagina Web con un paragrafo reso controllo HTML server.

```
<script runat="server" >
    sub Page_Load()
```

Loescher Editore - Vietata la vendita e la diffusione

```
            paragrafo.innerHTML="Prima pagina"
        End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
<h1>ASP.NET</h1>
<p id="paragrafo" runat="server" />
</body>
</html>
```

Figura 10.21 *Esecuzione dell'esempio*

Form e postback

I tag HTML di **input** devono essere contenuti in un elemento ***<form>*** con l'attributo ***runat="server"***. Nei form ASP.NET non è presente l'attributo *action*, infatti ogni pagina rimanda automaticamente a se stessa (**postback**). I dati inseriti in un form rimangono impostati tra un'elaborazione e l'altra. Si può controllare se è la prima volta che si elabora la pagina o no mediante la variabile booleana ***IsPostBack***.

Esempio 10.19

Uso di un form con l'attributo *runat="server"*.

```
<script runat="server" >
    sub Page_Load()
        paragrafo.innerHTML="Prima pagina"
    End Sub

    Sub Esegui(sender As Object, e As EventArgs)
        paragrafo2.innerHTML="Dopo il clic"
    End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
<h1>ASP.NET</h1>
<p id="paragrafo" runat="server" />
<form runat="server">
<input type="submit" runat="server" value="vai"
    onServerClick="Esegui"/>
</form>
<p id="paragrafo2" runat="server" />
</body>
</html>
```

Figura 10.22 *Esecuzione dell'esempio*

Viewstate

Le pagine forniscono la proprietà ***ViewState*** per mantenere lo stato della pagina (viewstate). Per default il viewstate è abilitato e viene mantenuto come campo nascosto aggiunto alla pagina.

Nota

Si può disabilitare il viewstate con l'attributo *EnableViewState="false"* della direttiva *@Page*. I dati inseriti in un form però rimangono impostati tra un'elaborazione e l'altra anche se il viewstate è disabilitato.

Loescher Editore - Vietata la vendita e la diffusione

Esempio 10.20

Postback di un form e uso del viewstate.

```
<script runat="server" >
    Sub Page_Load()
        If not IsPostBack Then
            ViewState("inizio")=DateTime.Now
        else
            Response.Write("Orario: " & DateTime.Now & " Hai iniziato a: " & ViewState("inizio"))
        End If
    End Sub

    Sub Esegui(sender As Object, e As EventArgs)
        messaggio.innerText = nome.value &" hai scelto il colore " & colore.value
    End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
<h1>ASP.NET</h1>
<form runat="server">
<input type="text" id="nome" runat="server" />
<select id="colore" runat="server">
<option>rosso</option>
<option>verde</option>
<option>blu</option>
</select>
<input type="submit" runat="server" value="vai"
    onServerClick="Esegui"/>
</form>
<p id="messaggio" runat="server" />
</body>
</html>
```

Figura 10.23 *Esecuzione dell'esempio*

Controlli Web server corrispondenti ai controlli HTML

Principali controlli Web corrispondenti ai controlli HTML:

Button	pulsante; l'attributo *Text* indica il testo sul pulsante;
CheckBox	casella di selezione; l'attributo *Text* indica il testo associato alla casella;
CheckBoxList	gruppo di caselle di selezione;
DropDownList	menu a discesa; le voci sono descritte da elementi *ListItem*;
HyperLink	collegamento ipertestuale; gli attributi *Text* o *ImageUrl* indicano il testo o l'immagine che fanno da collegamento; *NavigateUrl* specifica l'URL a cui punta il link;
Image	immagine; l'attributo *ImageUrl* indica il percorso dell'immagine; *AlternateText* permette di specificare un testo alternativo;
Login	fornisce un'interfaccia per il login dell'utente;
Label	etichetta; implementato per mezzo di tag **;
ListBox	casella di riepilogo; le voci sono descritte da elementi *ListItem*;
RadioButton	pulsante di selezione;
RadioButtonList	gruppo di pulsanti di selezione;
TextBox	casella di testo; l'attributo *Text* indica il testo contenuto nella casella.

Loescher Editore - Vietata la vendita e la diffusione

Esempio 10.21

Uso di controlli Web server.

```
<script runat="server" >
    Sub Esegui(sender As Object, e As EventArgs)
        scritta.Text=casella.Text
        colore.Text=selezione.selectedValue
    End Sub
</script>

<html>
<head>
<title>ASP.NET</title>
</head>
<body>
<h1>ASP.NET</h1>
<form runat="server">
<asp:TextBox id="casella" runat="server" />
<asp:DropDownList id="selezione" runat="server">
<asp:ListItem>rosso</asp:ListItem>
<asp:ListItem>verde</asp:ListItem>
<asp:ListItem>blu</asp:ListItem>
</asp:DropDownList>
<asp:Button runat="server" text="vai" onClick="Esegui"/>
</form>
<asp:Label id="scritta" runat="server" />
<asp:Label id="colore" runat="server" />
</body>
</html>
```

Figura 10.24 *Esecuzione dell'esempio*

Posting cross-page

È possibile effettuare il posting di dati a una pagina diversa (**posting cross-page**) indicando l'URL della pagina nella proprietà ***PostBackUrl*** del pulsante di invio *asp:Button*.
Nella pagina di destinazione è possibile fare riferimento alla pagina iniziale mediante il riferimento *PreviousPage*; per ottenere il valore dei campi inviati si utilizza il metodo ***FindControl()***.

Esempio 10.22

Posting cross page.

```
<html>
<head><title>ASP.NET</title></head>
<body>
<h1>ASP.NET</h1>
<form runat="server">
<input type="text" id="nome" runat="server" />
<select id="colore" runat="server">
<option>rosso</option>
<option>verde</option>
<option>blu</option>
</select>
```

Loescher Editore - Vietata la vendita e la diffusione

```
<asp:button runat="server" id="vai" Text="Vai" PostBackUrl="es22b.aspx" />
</form>
<p id="messaggio" runat="server" />
</body>
</html>
```

File es22b.aspx

```
<script runat="server" >
    Sub Page_Load()
        Dim nome As HtmlInputText
        Dim colore As HtmlSelect
        nome = PreviousPage.FindControl("nome")
        colore = PreviousPage.FindControl("colore")
        Response.Write(nome.value & " " & colore.value)
    End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
</body>
</html>
```

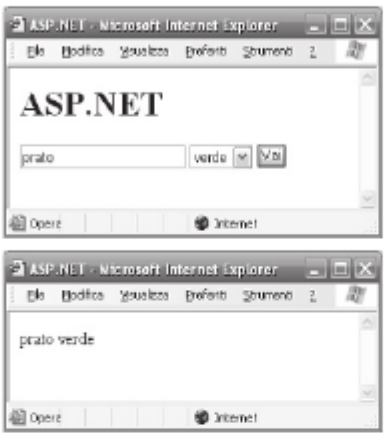


Figura 10.25 *Esecuzione dell'esempio*

Controlli di validazione

I controlli di validazione sono controlli Web server che permettono di eseguire facilmente validazioni di vario tipo sui dati.

Bisogna creare un controllo di validazione per ogni controllo che si vuole effettuare su ogni campo. La proprietà ***ControlToValidate*** specifica il campo da controllare.
Le proprietà ***Text*** e ***ErrorMessage*** specificano il messaggio da visualizzare in caso di errore; *Text* viene visualizzata nel punto in cui è inserito il controllo; *ErrorMessage* può essere usata al posto di *Text* ma, se sono presenti entrambe, viene utilizzata nel riepilogo degli errori che si può ottenere con ***ValidationSummary***.

I principali validatori sono:

RequiredFieldValidator	assicura la presenza del campo;
CompareValidator	confronta rispetto a un valore prefissato: si può confrontare con un valore (*ValueToCompare*) o con il contenuto di un altro campo (*ControlToCompare*); l'attributo *Operator* specifica il tipo di confronto (*GreaterThan*, *LessThan* o *Equal*) e l'attributo *Type* indica il tipo di dati;
RangeValidator	confronta rispetto a due limiti inferiore (*MinimumValue*) e superiore (*MaximumValue*); l'attributo *Type* indica il tipo di dati;
RegularExpressionValidator	confronta la corrispondenza con un pattern definito da una espressione regolare specificata in *ValidationExpression*.

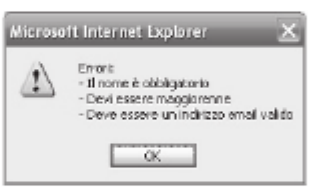
Il controllo ***ValidationSummary*** riassume tutti gli errori di validazione nella pagina (*ShowSummary="true"*) e/o in una finestra di dialogo (*ShowMessageBox="true"*).
L'attributo ***HeaderText*** indica il testo da visualizzare come intestazione del riepilogo e l'attributo ***DisplayMode*** il formato di output (per esempio il valore *BulletList* indica un elenco puntato).

Loescher Editore - Vietata la vendita e la diffusione

Esempio 10.23

Form con validazione dell'input.

```
<html>
<head><title>ASP.NET</title></head>
<body>
<form runat="server">
<table>
<tr><td>Nome:*</td>
<td><input type="text" id="nome" runat="server" size="50" />
<asp:RequiredFieldValidator runat="server" ControlToValidate="nome" Text="Manca" ErrorMessage="Il
nome è obbligatorio" /></td></tr>
<tr><td>Età:*</td>
<td><input type="text" id="eta" runat="server" size="50" />
<asp:RequiredFieldValidator runat="server" ControlToValidate="eta" Text="Manca" ErrorMessage="L'età è
obbligatoria" />
<asp:CompareValidator runat="server" ControlToValidate="eta"
ValueToCompare="18" Operator="GreaterThanEqual" Type="Integer" ErrorMessage="Devi essere
maggiorenne" /></td></tr>
<tr><td>Email:</td>
<td><input type="text" id="email" runat="server" size="50" />
<asp:RegularExpressionValidator runat="server" ControlToValidate="email"
ValidationExpression="[a-zA-Z_0-9.-]+\@[a-zA-Z_0-9.-]+\.\w+"
ErrorMessage="Deve essere un indirizzo email valido" /></td></tr>
</table>
<input type="submit" name="pulsanteInvio" runat="server" value="Invia">
<p>(*) I campi contrassegnati con l'asterisco sono obbligatori.</p>
<asp:ValidationSummary runat="server" ShowMessageBox="true"
ShowSummary="true" HeaderText="Errori:" DisplayMode="BulletList" />
</form>
</body>
</html>
```



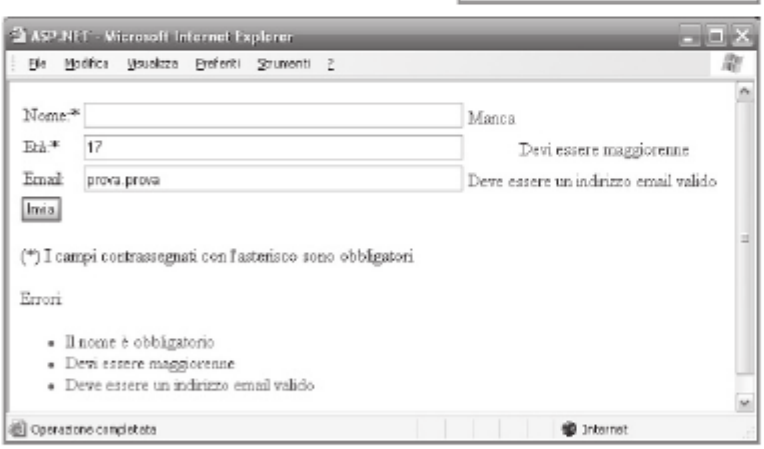


Figura 10.26 *Esecuzione dell'esempio*

Controllo Calendar

Il controllo ***Calendar*** visualizza un calendario mensile personalizzabile sia per quanto riguarda l'aspetto che le funzionalità. L'attributo *VisibleDate* imposta una data che deve essere visibile; l'attributo *SelectedData* specifica una data da evidenziare.

Loescher Editore - Vietata la vendita e la diffusione

Esempio 10.24

Calendario

```
<html>
head><title>ASP.NET</title></head>
<body>
<form runat="server">
<asp:Calendar runat="server" />
</form>
</body>
</html>
```

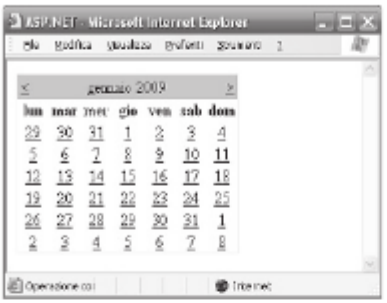


Figura 10.27 *Esecuzione dell'esempio*

10.11 ADO.NET

Nota

Per la connessione al database, l'uso degli oggetti *DbCommand*, *DataSet* e *DataAdapter* e il data binding si veda il paragrafo 9.2.

DbCommand

Esempio 10.25

Lettura con *DataReader*.

```
<% @Import NameSpace="System.Data.OleDb" %>

<script runat="server">
Sub Page_Load()
Dim origine As OleDbConnection
Dim selezione As OleDbCommand
Dim dati As OleDbDataReader
origine = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
Server.MapPath("cap.mdb"))
origine.Open()
selezione = New OleDbCommand("SELECT * FROM elenco", origine)
dati = selezione.ExecuteReader()
While dati.Read()
Response.Write(dati.Item("citta") & " " & dati.Item("cap") & "<br />")
End While
origine.close()
End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
</body>
</html>
```

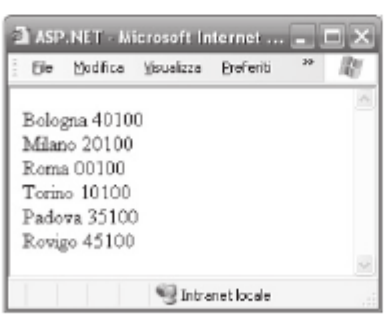


Figura 10.28 *Esecuzione dell'esempio*

Loescher Editore - Vietata la vendita e la diffusione

Esempio 10.26

Inserimento di dati in modalità connessa.

```
<% @Import NameSpace="System.Data.OleDb" %>

<script runat="server">
Sub esegui(s As Object, e AS EventArgs)
Dim origine As OleDbConnection
Dim comando As OleDbCommand
origine = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
Server.MapPath("cap.mdb"))
origine.Open()
comando = New OleDbCommand("INSERT INTO elenco (citta, cap) VALUES (città.Text, cap.Text)", origine)
comando.Parameters.AddWithValue("@città", città.Text)
comando.Parameters.AddWithValue("@cap", cap.Text)
comando.ExecuteNonQuery()
origine.close()
End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
<form runat="server">
<table>
<tr><td>Città:</td><td><asp:textbox id="città" runat="server" /></td></tr>
<tr><td>CAP:</td><td><asp:textbox id="cap" runat="server" /></td></tr>
</table>
<br/>
<asp:Button onClick="esegui" Text="Inserisci" runat="server" />
</form>
</body>
</html>
```

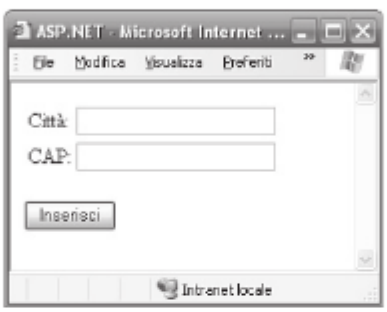


Figura 10.29 *Esecuzione dell'esempio*

DataSet e DataAdapter

Esempio 10.27

Lettura con *DataSet*.

```
<% @Import NameSpace="System.Data.OleDb" %>
<% @Import NameSpace="System.Data" %>

<script runat="server">
Sub Page_Load()
Dim origine As OleDbConnection
Dim selezione As OleDbDataAdapter
Dim dati As DataSet
Dim lettore As DataTableReader
origine = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
Server.MapPath("cap.mdb"))
origine.Open()
selezione = New OleDbDataAdapter("SELECT * FROM elenco", origine)
```

Loescher Editore - Vietata la vendita e la diffusione

```
dati = new DataSet()
selezione.Fill(dati, "elenco")
origine.close()
lettore = dati.CreateDataReader()
While lettore.Read()
Response.Write(lettore(0) & " " & lettore(1) & "<br />")
End While
lettore.close()
End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
</body>
</html>
```

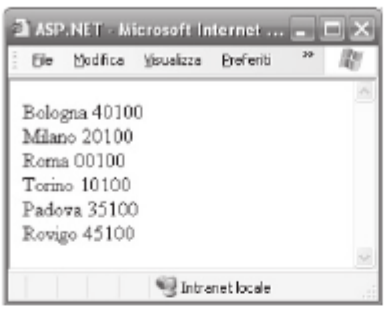


Figura 10.30 *Esecuzione dell'esempio*

Data binding

Il **data binding** (associazione dati) in ASP.NET può essere realizzato usando:

- espressioni di data binding;
- controlli di data bound (binding basato su data source);
- controlli data source.

Espressioni di data binding

La sintassi di una **espressione di data binding** è:

```
<%# espressione %>
```

Il binding viene effettuato solo quando viene richiamato il metodo ***DataBind()*** per tutta la pagina o su un singolo elemento.

Esempio 10.28

Data binding con espressioni.

```
<script runat="server">
Dim mess As String = "Scegli un colore"

Sub Page_Load()
DataBind()
End Sub

Sub Cambia(sender As Object, e As EventArgs)
colore.DataBind()
End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
<form runat="server">
<asp:Label runat="server" id="messaggio" Text='<%# mess %>' /><br/>
<asp:ListBox runat="server" id="colori" AutoPostBack="true" onSelectedIndexChanged="Cambia">
```

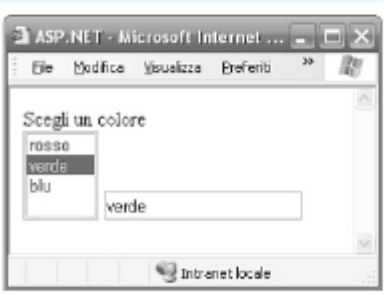


Figura 10.31 *Esecuzione dell'esempio*

Loescher Editore - Vietata la vendita e la diffusione

```
<asp:ListItem value="rosso" selected="true">rosso</asp:ListItem>
<asp:ListItem value="verde">verde</asp:ListItem>
<asp:ListItem value="blu">blu</asp:ListItem>
</asp:ListBox>
<asp:TextBox runat="server" id="colore" Text='<%# colori.SelectedItem.Text %>' />
</form>
</body>
</html>
```

La proprietà ***AutoPostBack*** permette ai controlli Web di inviare immediatamente i dati appena si verifica qualsiasi evento.

Controlli data bound

I **controlli data bound** sono controlli che dispongono della proprietà ***DataSource*** per effettuare il collegamento ai dati.

Nota

Dal nome della proprietà *DataSource* deriva il termine "binding basato su data source" usato per questa modalità.

Sono controlli data bound:

- gli elementi HTML con capacità di collegare dati (come *CheckBoxList*, *DropDownList*, *ListBox* e *RadioButtonList*);
- i controlli iterativi progettati specificamente come *Repeater*, *DataList*, *DataGrid* e *GridView*, *DetailsView* e *FormView*.

I controlli data bound espongono le proprietà relative ai dati; le principali **proprietà di data binding** sono:

DataSource	specifica la sorgente a cui il controllo è legato;
DataSourceId	id del componente data source (mutuamente esclusiva con *DataSource*); permette di collegare i nuovi controlli data bound ai controlli data source;
DataMember	nome della collezione di dati da estrarre quando si effettua il data binding a un data source (per esempio una tabella in un *DataSet*);
DataTextField	specifica quale proprietà di un elemento data bound usare per definire il testo da visualizzare (per esempio un campo di una tabella).

I controlli iterativi forniscono un meccanismo basato su template per creare interfacce utente con layout personalizzato: accettano una sorgente dati, iterano sugli elementi della sorgente e applicano a ogni riga dei template HTML definiti dal programmatore.

Elementi HTML

Esempio 10.29

Assegnazione dei dati di una colonna di una tabella a un elenco a discesa.

```
<% @Import NameSpace="System.Data.OleDb" %>

<script runat="server">
Sub Page_Load()
if Not IsPostBack Then
Dim origine As OleDbConnection
Dim selezione As OleDbCommand
```

```
Dim dati As OleDbDataReader
origine = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
Server.MapPath("cap.mdb"))
origine.Open()
selezione = New OleDbCommand("SELECT * FROM elenco", origine)
dati = selezione.ExecuteReader()
scelta.Datasource = dati
scelta.DataTextField = "citta"
scelta.DataBind()
origine.close()
End If
End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
<form runat="server">
<asp:DropDownList id="scelta" runat="server" />
</form>
</body>
</html>
```

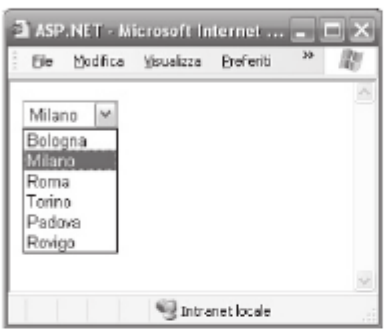


Figura 10.32 *Esecuzione dell'esempio*

Nota Invece di *DropDownList* si può usare *ListBox*, *RadioButtonList* o *CheckBoxList*.

Repeater

Il controllo ***Repeater*** visualizza i dati usando layout specificati dall'utente mediante template. Tutte le informazioni di formattazione e di layout devono essere dichiarate esplicitamente.
I template utilizzabili sono:

HeaderTemplate	intestazione;
ItemTemplate	righe;
AlternatingItemTemplate	righe di valore dispari per definire stili diversi per righe alternate;
FooterTemplate	fine dell'elenco, chiusura.

Il controllo *Repeater* consente solo la visualizzazione dei dati; non si può usare per effettuare modifiche.
Per ottenere i dati relativi a un campo del record corrente si possono usare, in alternativa:

```
Container.DataItem("nomeCampo")

Eval("nomeCampo")
```

Nota È disponibile anche il controllo simile *DataList* che, rispetto a *Repeater*, permette il rendering direzionale cioè gli elementi possono essere disposti su più colonne, inoltre offre un supporto per la selezione e l'editing.

Esempio 10.30

Visualizzazione con *Repeater*.

```
<% @Import NameSpace="System.Data.OleDb" %>
```

```
<script runat="server">
Sub Page_Load()
Dim origine As OleDbConnection
Dim selezione As OleDbCommand
Dim dati As OleDbDataReader
origine = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
Server.MapPath("cap.mdb"))
origine.Open()
selezione = New OleDbCommand("SELECT * FROM elenco", origine)
dati = selezione.ExecuteReader()
rep.Datasource = dati
rep.DataBind()
origine.close()
End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
<form runat="server">
<asp:Repeater id="rep" runat="server">
    <HeaderTemplate>
    <table border="1">
    </HeaderTemplate>
    <ItemTemplate>
    <tr>
        <td><%#Container.DataItem("citta") %></td>
        <td><%#Container.DataItem("cap") %></td>
    </tr>
    </ItemTemplate>
    <FooterTemplate>
    </table>
    </FooterTemplate>
</asp:Repeater>
</form>
</body>
</html>
```

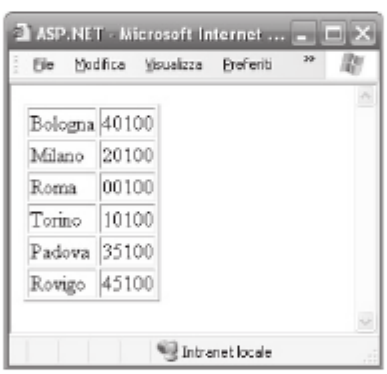


Figura 10.33 *Esecuzione dell'esempio*

GRIDVIEW

GridView realizza automaticamente una tabella con i dati ed è molto personalizzabile.

Nota

GridView è stato introdotto in ASP.NET 2.0; è ancora disponibile, ma ormai superato il precedente controllo *DataGrid*.

ESEMPIO 10.31

Visualizzazione con *GridView*.

```
% @Import NameSpace="System.Data.OleDb" %>

<script runat="server">
Sub Page_Load()
```

```
Dim origine As OleDbConnection
Dim selezione As OleDbCommand
Dim dati As OleDbDataReader
origine = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
Server.MapPath("cap.mdb"))
origine.Open()
selezione = New OleDbCommand("SELECT * FROM elenco", origine)
dati = selezione.ExecuteReader()
gv.Datasource = dati
gv.DataBind()
origine.close()
End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
<form runat="server">
<asp:GridView id="gv" runat="server">
</asp:GridView>
</form>
</body>
</html>
```

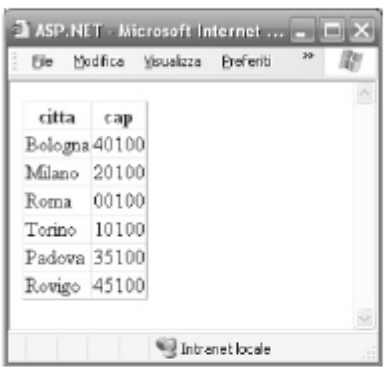


Figura 10.34 *Esecuzione dell'esempio*

DetailsView e FormView

DetailsView e ***FormView*** visualizzano un singolo record per volta; possono fornire pulsanti per navigare tra i record e permettono anche di aggiornare il database. Esprimono tutta la loro potenza quando sono collegati a controlli data source.

DetailsView utilizza un'interfaccia generata automaticamente mentre ***FormView*** permette di personalizzare l'interfaccia creando un form con layout qualsiasi.

Controlli data source

I **controlli data source**, introdotti in ASP.NET 2.0, definiscono una sorgente dati in modo dichiarativo e offrono capacità avanzate (come ordinamento, paginazione, aggiornamento) senza richiedere codice.

Nota

I controlli data source nascondono la complessità del data binding manuale: forniscono i dati ai controlli data bound e li supportano nell'esecuzione delle operazioni comuni.

Sono disponibili vari tipi di controlli data source:

AccessDataSource	connessione a database Microsoft Access;
SqlDataSource	connessione a database SQL Server;
XmlDataSource	connessione a file XML.

Per collegare un controllo data bound a un controllo data source si usa la proprietà ***DataSourceId*** dei controlli data bound.
Per eseguire le operazioni comuni non è necessario scrivere codice: basta impostare in modo dichiarativo le proprietà dei controlli data bound e le istruzioni da usare nei controlli data source.

Loescher Editore - Vietata la vendita e la diffusione

Per la connessione a un database, le proprietà dei controlli data source che permettono di impostare i comandi SQL da usare rispettivamente per recuperare i dati, inserire nuove righe, aggiornare le righe e cancellare le righe sono ***SelectCommand***, ***InsertCommand***, ***UpdateCommand*** e ***DeleteCommand***. Le operazioni avvengono nel database.

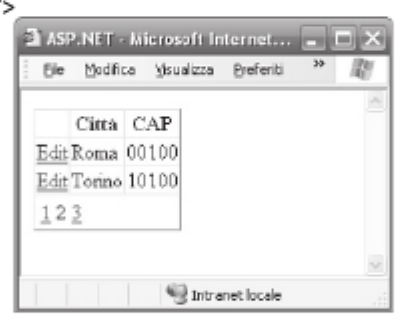
Nei controlli data bound:

- *AllowPaging* e *PageSize* impostano proprietà di paginazione;
- *AutoGenerateEditButton="true"* crea automaticamente una colonna di pulsanti per modificare il record;
- se è impostato *AutoGenerateColumns="false"* le colonne possono essere definite manualmente con elementi *BoundField*.

Esempio 10.32

GridView con data source *AccessDataSource*.

```
<html>
<head><title>ASP.NET</title></head>
<body>
<form runat="server">
<asp:AccessDataSource runat="server" id="db" DataFile="cap.mdb"
SelectCommand = "SELECT * FROM elenco"
UpdateCommand = "UPDATE elenco SET cap=@cap WHERE citta=@citta>"
</asp:AccessDataSource>
<asp:GridView id="gv" runat="server" DataSourceId="db" AutoGenerateColumns="false"
AllowPaging="true" PageSize="2" AutoGenerateEditButton="true">
<columns>
<asp:BoundField DataField="citta" HeaderText="Città"/>
<asp:BoundField DataField="cap" HeaderText="CAP"/>
</columns>
</asp:GridView>
</form>
</body>
</html>
```



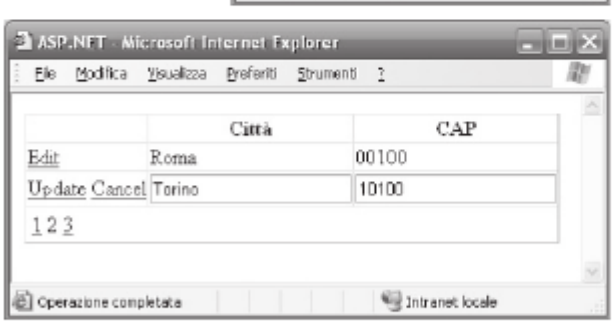


Figura 10.35 *Esecuzione dell'esempio*

Esempio 10.33

Visualizzazione con *DetailsView* e data source *AccessDataSource*.

```
<html>
<head><title>ASP.NET</title></head>
<body>
```

```
<form runat="server">
<asp:AccessDataSource runat="server" id="db" DataFile="cap.mdb"
SelectCommand="SELECT * FROM elenco">
</asp:AccessDataSource>
<asp:DetailsView id="dv" runat="server" DataSourceId="db" AutoGenerateRows="false" AllowPaging="true"
HeaderText="Codici">
<PagerSettings Mode="NextPreviousFirstLast" />
<Fields>
<asp:BoundField DataField="citta" HeaderText="Città"/>
<asp:BoundField DataField="cap" HeaderText="CAP"/>
</Fields>
</asp:DetailsView>
</form>
</body>
</html>
```

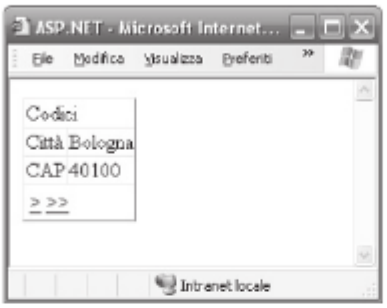


Figura 10.36
Esecuzione dell'esempio

10.12 SICUREZZA DELLE APPLICAZIONI SERVER SIDE

Le applicazioni server side, se scritte male, possono presentare **vulnerabilità** che si prestano ad **exploit**, in particolare del tipo **SQL injection**.

VULNERABILITÀ ED EXPLOIT

Per **vulnerabilità** si intende un difetto del codice, dovuto a qualche errore o superficialità nella realizzazione del programma, che si presta ad attacchi.
Spesso la vulnerabilità è dovuta alla mancata **validazione dell'input** nel programma; non si deve lasciare all'utente la possibilità di inserire qualcosa che non è permesso.

Gli **exploit** sono attacchi che sfruttano le vulnerabilità del software.
Le vulnerabilità in genere possono essere sfruttate solo con procedimenti abbastanza complessi da individuare; una volta che sono stati individuati e pubblicizzati però possono essere usati anche da persone non eccessivamente competenti.

SQL INJECTION

Con un attacco di tipo ***SQL injection*** si riesce a far eseguire comandi SQL arbitrari e non autorizzati ad una applicazione che esegue una query SQL costruita in base a input passati da un utente.

Nota
Si ribadisce che questi attacchi sono possibili solo in presenza di codice vulnerabile cioè non realizzato in modo corretto e sicuro.

La possibilità di eseguire l'attacco dipende dalla mancata **validazione dell'input** da parte di chi ha scritto il codice.
Per evitare un attacco SQL injection si può controllare che vengano inseriti solo i caratteri strettamente richiesti o almeno sostituire i caratteri ' e " con le sequenze di escape \' e \".

Nota
Alcuni server Web fanno automaticamente la sostituzione con le sequenze di escape (per esempio in PHP l'escape è controllato dalla direttiva *magic_quotes_gpc* nel file di configurazione *php.ini*, che per default è attiva.

Loescher Editore - Vietata la vendita e la diffusione

Esempio 10.34 - SQL injection

Un'istruzione che accede a una tabella di utenti:

in PHP

```
$risultato=mysql_query(select * from utenti where nome = '$nome' and password = '$password');
```

in JSP

```
String nome = request.getParameter("nome");
String password = request.getParameter("password");
ResultSet rs = st.executeQuery("SELECT * from utenti where nome='" + nome +"' and password='" +
password +"'");
```

può essere usata per ottenere l'autorizzazione senza conoscere la password; per esempio inserendo come nome *admin* e come password:

1' or '1'='1

l'istruzione diventa:

select * from utenti where nome = 'admin' and password = '1' or '1'='1'

rendendo sempre vera la condizione ed eludendo il controllo della password.

L'istruzione potrebbe essere usata addirittura per inserire nel database un nuovo utente, che poi può essere usato per accedere al sistema; inserendo un nome qualsiasi (per esempio *prova*) e come password:

prova'; insert into utenti (nome, password) values ('pippo', 'pippo'); - -

l'istruzione diventa:

select * from utenti where nome = 'prova' and password = 'prova'; insert into utenti (nome, password) values ('pippo', 'pippo'); --'

che corrisponde a due istruzioni; la prima cerca i dati (anche se non li trova), la seconda inserisce il nuovo utente nella tabella; i trattini indicano l'inizio di un commento (perché non ci siano errori di sintassi a causa degli apici che delimitano le stringhe).

Nota

Se il DBMS non supporta gli statement multipli viene prodotto soltanto un messaggio di errore.

Mediazione incompleta ed errori tempo di controllo/tempo di utilizzo

Altri errori nella scrittura dei programmi che si prestano ad attacchi di vario tipo sono:

Mediazione incompleta

è una situazione in cui dei dati sensibili si trovano in una condizione esposta non controllata, cioè i valori dei dati non sono completamente mediati; per esempio quando si passano dei parametri mediante un URL come nell'istruzione:

URL?param1=val1&param2=val2

i valori dei parametri potrebbero essere modificati dall'utente; per esempio se tramite un parametro di questo tipo viene passato il prezzo di un articolo acquistato l'utente potrebbe modificarlo con un valore inferiore;

Errori tempo di controllo/tempo di utilizzo

è una situazione in cui viene effettuato un controllo in un certo momento su chi accede, ma successivamente viene eseguita un'azione diversa senza che il controllo venga ripetuto.

Loescher Editore - Vietata la vendita e la diffusione

VERIFICA

QUESTIONARIO

1. Come funziona il protocollo HTTP?
2. Che cosa si intende dicendo che il protocollo HTTP è privo di stati?
3. Che cosa descrive l'interfaccia CGI?
4. Che cosa sono le variabili d'ambiente?
5. Come vengono codificati i dati inviati all'applicazione server side?
6. Qual è la differenza tra i metodi *GET* e *POST*?
7. Quali sono le principali tecnologie server side?
8. Che cosa sono e a che cosa servono i cookie?
9. Che cosa sono le sessioni e come vengono gestite?
10. Com'è costituita un'architettura 3-tier per eseguire elaborazioni su database con applicazioni server side?
11. Qual è la differenza fra hosting e housing?
12. Con quale servizio si possono pubblicare le pagine Web?
13. Che cosa è necessario per rendere disponibile un server Web in una intranet o da Internet?
14. A che cosa serve il file *php.ini*?
15. Che cosa sono gli array associativi in PHP?
16. Che cos'è una stringa in PHP?
17. A che cosa servono gli array superglobali *$_GET* e *$_POST* in PHP?
18. A che cosa serve la variabile *$_SERVER['PHP_SELF']* in PHP?
19. Che cos'è un servlet?
20. Che relazione c'è tra pagine JSP e servlet?
21. Quali sono le analogie e le differenze tra pagine JSP e servlet?
22. In che cosa consistono le fasi di page translation e page compilation?
23. Che cosa succede al momento della richiesta di una pagina JSP?
24. Che cosa succede se una pagina JSP contiene errori?
25. Che cos'è la tecnologia JavaBeans?
26. Quali sono gli oggetti predefiniti che si possono usare nelle pagine JSP?
27. Come si possono gestire gli errori nelle pagine JSP?
28. Quali linguaggi si utilizzano per scrivere il codice nelle pagine ASP?
29. Quali sono gli oggetti predefiniti che si possono usare nelle pagine ASP?
30. A che cosa servono le collezioni *Request.QueryString* e *Request.Form* in ASP?
31. Che cos'è il framework .NET?
32. Quali linguaggi si utilizzano per scrivere il codice nelle pagine ASP.NET?
33. Qual è il paradigma di programmazione usato nelle pagine ASP.NET?
34. Che cosa succede al momento della richiesta di una pagina ASP.NET?
35. A che cosa servono i file *machine.config* e *web.config*?
36. Che cosa sono i controlli HTML server e i controlli Web server?
37. Che cos'è il postback?
38. Che cosa sono i controlli data bound?
39. Che cosa sono i controlli data source?
40. Che cosa si intende per vulnerabilità di un sistema?
41. Che cosa sono gli exploit?

Loescher Editore - Vietata la vendita e la diffusione

42. In che cosa consiste un attacco SQL injection?
43. Che cosa si intende per mediazione incompleta?

Esercizi

1 ..... Visualizzare i dati di una tabella di un database in forma tabellare e fare in modo che sia possibile ordinare i dati in base ai valori di una colonna, facendo clic sull'intestazione della colonna.

2 ..... Modificare l'esercizio 1 in modo che la tabella di visualizzazione dei dati presenti per ogni riga un collegamento ipertestuale (o un pulsante) per la variazione del record.

3 ..... Modificare l'esercizio 1 in modo che i dati vengano visualizzati tramite caselle di testo che permettano di effettuare direttamente la variazione.

PROPOSTE DI LAVORO

Progettare una interfaccia per eseguire via Web tramite pagine PHP, JSP, ASP o ASP.NET le seguenti elaborazioni.

1 ..... Realizzare una applicazione per giocare a dadi col computer; ognuno lancia due dadi; vince chi ottiene lo stesso risultato sui due dadi o chi fa il punteggio più alto con il lancio; visualizzare i risultati dei due lanci e l'indicazione del vincitore.

2 ..... Realizzare un modulo che permetta di inserire due operandi e un simbolo di operazione; la pagina che elabora i dati deve visualizzare i dati inseriti e chiedere conferma prima di effettuare l'operazione.

3 ..... Realizzare un modulo che permetta di inserire una data ed effettuarne la validazione in una pagina server side.

4 ..... Realizzare un modulo che permetta di inserire due operandi e di scegliere una della quattro operazioni da un menu a discesa; visualizzare il risultato nello stesso modulo, ancora impostato con i dati scelti.

5 ..... Realizzare un modulo che proponga una domanda con varie risposte di cui una sola esatta; la risposta può essere scelta mediante pulsanti di selezione. Rivisualizzare il modulo con il pulsante selezionato dall'utente e l'indicazione se la risposta è esatta o errata.

6 ..... Realizzare un'applicazione che mostri una pagina con frame o senza frame in base a un cookie.

7 ..... Realizzare una pagina che contenga tre sezioni (per esempio Musica, Cinema e Sport); per ogni sezione sia disponibile un collegamento che permetta di scegliere se visualizzare o meno la sezione (on/off); mantenere le scelte fatte con dei cookie in modo che alle visualizzazioni successive vengano presentate solo le sezioni scelte.

8 ..... Realizzare una pagina che presenti il menu di un ristorante; accanto ad ogni piatto ci sia un collegamento per aggiungere il piatto all'ordinazione (ogni piatto scelto va registrato come attributo di sessione). Permettere in ogni momento di vedere i piatti ordinati e di togliere un piatto dall'ordinazione.

Loescher Editore - Vietata la vendita e la diffusione

9 ..... Realizzare un carrello della spesa; l'applicazione deve visualizzare un elenco di articoli (per esempio libri) e permettere di aggiungere ciascun articolo al carrello. In ogni momento deve essere possibile vedere il contenuto del carrello e modificarlo.

10 ..... Contare le visite effettuate a ciascuna pagina del sito da un utente durante la sessione. Visualizzare le informazioni quando viene selezionato un apposito collegamento.

11 ..... Contare il numero degli utenti che accedono al sito (cioè il numero delle sessioni attivate) fino allo spegnimento del server. Visualizzare le informazioni quando viene selezionato un apposito collegamento.

12 ..... Richiedere ad ogni utente che visita il sito di inserire il proprio nome e di scegliere il nome di un animale, di una pianta e di un colore tra alcuni proposti. Mantenere le informazioni inserite fino allo spegnimento del server e quando viene selezionato un apposito collegamento mostrare la classifica delle scelte effettuate e il nome degli utenti, se ce ne sono, che hanno effettuato le tre scelte prime in graduatoria.

Progettare il database e una interfaccia per eseguire via Web tramite pagine PHP, JSP, ASP o ASP.NET le seguenti elaborazioni.

13 ..... *Prodotti* (codice prodotto, descrizione, prezzo); operazioni possibili:
- visualizzare i dati presenti nella tabella;
- cercare la descrizione e il prezzo di un prodotto dato il codice;
- cercare il prezzo di un prodotto data la descrizione;
- inserire i dati di un nuovo prodotto;
- modificare la descrizione o il prezzo di un prodotto;
- visualizzare i record in successione, uno alla volta.

14 ..... *Anagrafiche* degli studenti di una classe (codice, nome, indirizzo, data di nascita ecc.); operazioni possibili:
- inserire i dati di uno studente;
- visualizzare i dati di tutti gli studenti;
- visualizzare i dati di uno studente richiesto, dato il codice o il nome;
- visualizzare i dati di tutti gli studenti il cui nome inizia con una data lettera;
- visualizzare l'elenco degli anni in cui sono nati gli studenti;
- modificare l'indirizzo di uno studente richiesto, dato il codice;
- cancellare i dati di uno studente, dato il codice.

15 ..... Database *Geografia* con stati e capitali; operazioni possibili:
- inserimento di dati;
- visualizzazione dei dati;
- ricerca della capitale, dato lo stato;
- ricerca dello stato, data la capitale.

16 ..... Database *Agenda* di appuntamenti; un appuntamento ha una data, un'ora di inizio e una di fine e una descrizione; operazioni possibili:
- aggiungere appuntamenti;
- eliminare appuntamenti annullati;
- stampare tutti gli appuntamenti;
- stampare l'elenco di appuntamenti per una giornata;
- stampare l'elenco di appuntamenti in un intervallo di ore.

17 ..... *Forum* di messaggi (gruppi di discussione consultabili via Web); per poter inviare un messaggio a un gruppo un utente deve essere registrato; operazioni possibili:
- vedere l'elenco dei gruppi esistenti (argomenti);
- vedere l'elenco delle intestazioni dei messaggi di un particolare gruppo;

Loescher Editore - Vietata la vendita e la diffusione

- vedere il testo di un particolare messaggio;
- eseguire la registrazione;
- inviare un messaggio a un gruppo.

18 ..... *Prodotti* raggruppati per categorie; operazioni possibili:
- visualizzare le categorie di prodotti;
- visualizzare i dati dei prodotti di una determinata categoria (codice, nome e prezzo);
- visualizzare i dati di uno specifico prodotto (categoria del prodotto, codice, nome e prezzo, descrizione più dettagliata e immagine del prodotto);
- solo per l'amministratore, aggiornare i dati.

19 ..... Database di immagini (descrizione e file dell'immagine); operazioni possibili:
- aggiornare il database;
- visualizzare l'elenco delle immagini con la relativa descrizione;
- visualizzare le immagini una alla volta con pulsanti per scorrere avanti e indietro e andare all'inizio e alla fine;
- visualizzare automaticamente le immagini una dopo l'altra producendo un'animazione con pulsanti per avvio, pausa e stop.

20 ..... Database di banner pubblicitari (immagine, descrizione e link a cui andare quando si fa clic sull'immagine); operazioni possibili:
- visualizzare un banner in modo random in una finestra all'apertura di una pagina;
- visualizzare i banner uno dopo l'altro in sequenza, all'infinito;
- fare clic su un banner per essere trasferiti alla pagina corrispondente.

21 ..... Database con parole e definizioni di un glossario; operazioni possibili:
- visualizzare l'elenco delle definizioni;
- visualizzare l'elenco delle definizioni che iniziano con una lettera specificata;
- cercare la definizione di una parola specificata;
- solo per l'amministratore, aggiornare il database.

22 ..... Rubrica telefonica con nomi, indirizzi, vari numeri di telefono e indirizzi email; operazioni possibili:
- visualizzare l'elenco dei dati;
- effettuare operazioni di aggiornamento;
- cercare i dati di una persona specificata;
- inviare un messaggio email a uno degli indirizzi email di una persona.

23 ..... Database con nomi e password di utenti; operazioni possibili:
- effettuare una registrazione per ottenere nome e password;
- accedere a pagine protette solo dopo aver effettuato correttamente il login.

24 ..... Gestione delle riviste di una casa editrice; operazioni possibili:
- visualizzare l'elenco delle riviste disponibili;
- visualizzare la descrizione relativa ad una particolare rivista;
- effettuare una registrazione per ottenere un account da usare per sottoscrivere gli abbonamenti;
- sottoscrivere un nuovo abbonamento se si è registrati;
- disdire un precedente abbonamento.

25 ..... Vocabolario inglese/italiano e viceversa; operazioni possibili:
- visualizzare l'elenco di tutte le parole con la relativa traduzione;
- visualizzare l'elenco delle parole che iniziano con una lettera specificata;
- visualizzare la traduzione di una parola;
- presentare una parola alla volta, in una sola lingua, e richiedere all'utente la traduzione, contando le risposte esatte.

Loescher Editore - Vietata la vendita e la diffusione

26 ..... Database con domande e risposte di un test, con la possibilità di scegliere più risposte per ogni domanda; operazioni possibili:
- eseguire il test;
- visualizzare una statistica sulle risposte scelte.

27 ..... Database con domande e risposte per un test del tipo vero/falso; operazioni possibili:
- eseguire il test, una domanda per volta o tutte insieme, eventualmente con la possibilità di correggersi e alla fine vedere l'esito del test;
- per l'insegnante, vedere i risultati dei test con statistiche sulle risposte;
- per l'insegnante, inserire un nuovo test.

28 ..... Database con domande e risposte per un test a risposta multipla (domande con varie risposte per ogni domanda e identificazione della risposta esatta); operazioni possibili:
- eseguire il test, una domanda per volta o tutte insieme, eventualmente con la possibilità di correggersi e alla fine vedere l'esito del test;
- per l'insegnante, vedere i risultati dei test con statistiche sulle risposte;
- per l'insegnante, inserire un nuovo test.

29 ..... Gestione di un database con articoli di una rivista (titolo, autore, data, testo dell'articolo memorizzato in un file) con possibilità di aggiornare il database e di effettuare ricerche di vario tipo.

30 ..... Realizzare per un sito:
- contatore delle visite al sito;
- contatore delle visite a ciascuna pagina;
- contatore delle visite a ciascuna pagina per mese;
- statistiche sui siti da cui si è arrivati alla pagina e sui browser utilizzati.

31 ..... Partendo dalla proposta di lavoro 6 del capitolo 9 si vuole tenere traccia delle sessioni che gli utenti hanno col motore di ricerca; in una sessione di ricerca l'utente cerca un termine, ottiene l'elenco delle pagine correlate e legge alcune di queste pagine; per ogni sessione interessa il giorno in cui si è svolta e il termine di ricerca usato. La sessione può essere eseguita da un utente anonimo o da un utente registrato; per ogni utente registrato (di cui sono noti username, password e indirizzo email) si vuole memorizzare l'elenco delle pagine lette.

32 ..... Realizzare una conferenza stampa su Web; ci sono più intervistatori e più intervistati; ogni intervistatore può porre domande a un intervistato e vede le risposte; ogni intervistato può rispondere alle domande di cui vede l'elenco; deve sempre essere possibile vedere l'elenco delle domande con le relative risposte.

33 ..... Gestione delle iscrizioni a una scuola.

34 ..... Gestione del registro dei voti di una o più classi.

35 ..... Gestione di una biblioteca con prestiti e prenotazione.

36 ..... Gestione degli acquisti online per una azienda (con gestione del catalogo e del carrello della spesa).

37 ..... Gestione di vendite/acquisto di libri usati da parte degli studenti.

38 ..... Gestione di una agenzia immobiliare.

39 ..... Gestione di una agenzia viaggi.

Loescher Editore - Vietata la vendita e la diffusione

Modulo 5

XML

11 XML e XSL

12 Elaborazione di documenti

Loescher Editore - Vietata la vendita e la diffusione

11 XML e XSL

Prerequisiti

- Funzionamento del servizio Web e linguaggio HTML

Obiettivi

Conoscenze

- Linguaggi SGML e XML
- Documenti XML
- Spazi dei nomi (namespace)
- XLink e XPointer
- DTD
- Linguaggio XSL
- XPath

Abilità/capacità

- Scrivere documenti XML
- Utilizzare collegamenti ipertestuali nei documenti XML
- Capire le DTD per i documenti XML
- Scrivere fogli stile XSL e applicarli a documenti XML

Loescher Editore - Vietata la vendita e la diffusione

11.1 INTRODUZIONE

XML (eXtensible Markup Language) è un **linguaggio di markup** come l'HTML ma, rispetto all'HTML, permette di definire più chiaramente la struttura del documento e di creare nuovi tag.

Il progetto XML prevede anche l'utilizzo di **fogli stile** per la presentazione del documento (CSS - Cascading Style Sheets e **XSL** - EXtensible Style Language) e la definizione di un apposito linguaggio per l'utilizzo di **collegamenti ipertestuali** (**XLink**).

Per visualizzare i dati di un documento XML si può usare un browser che supporti XML o una applicazione realizzata appositamente.
In ogni caso serve un **parser** che analizzi il documento e ne renda disponibili le informazioni.
Per visualizzare un documento nel **browser** si possono usare i fogli stile **XSL**.

Per fare in modo che altri creino documenti XML dello stesso tipo e che possano quindi usare le applicazioni realizzate bisogna definire una DTD; la **DTD** specifica quali tag e attributi si possono usare nel documento.

11.2 IL LINGUAGGIO XML

Un linguaggio di **markup** o linguaggio di contrassegno è un linguaggio che permette di aggiungere mediante marcatori (**tag**) informazioni a un documento; il documento risulta costituito da contenuto e da sistema di contrassegno.

SGML (Standard Generalized Markup Language), nato nel 1986 e adottato dall'ISO (International Standards Organization), è un **metalinguaggio** per la creazione di linguaggi di markup.
Il linguaggio che si vuole creare viene definito in una tabella chiamata **DTD** (Document Type Definition); nella DTD si definiscono i tag che si possono usare e le relazioni tra i tag.

Nota

HTML è un linguaggio di markup definito con SGML.

XML (eXtensible Markup Language) è un sottoinsieme semplificato di SGML pensato appositamente per la creazione di documenti su Web.

L'XML permette di specificare un linguaggio di markup mediante una DTD, ma permette anche di creare documenti che non utilizzano DTD.

La **DTD** definisce i tag del linguaggio, gli attributi, e il modo in cui si possono utilizzare.
Un documento che rispetta le regole di una DTD si dice **valido** rispetto alla DTD.

Un documento che non utilizza una DTD, ma rispetta le regole sintattiche dei documenti XML, si dice **ben formato** (well formed).

In un documento XML i **tag** sono racchiusi tra parentesi angolari (< >); ogni tag, aperto e chiuso, identifica un elemento.
L'XML permette di descrivere chiaramente la **struttura** del documento, in modo gerarchico.
La formattazione è compito di sistemi di contrassegno procedurali esterni (**fogli stile**).

Gli **usi** principali dell'XML riguardano:

- la descrizione dei **documenti** (in alternativa all'HTML);
  i documenti XML, poiché la struttura è definita chiaramente, permettono di eseguire elaborazioni più raffinate e facilitano l'indicizzazione da parte dei motori di ricerca, che possono compiere ricerche più specifiche.

Loescher Editore - Vietata la vendita e la diffusione

> Si possono per esempio creare automaticamente indici o sommari o eseguire ricerche su particolari elementi del documento o sui tag stessi.

- la descrizione dei **dati** (in alternativa ai database);
  l'XML costituisce un modo standard di descrivere i dati invece di utilizzare i formati proprietari dei vari DBMS.

L'utilizzo di **fogli stile esterni** per la presentazione del documento permette di creare dinamicamente viste diverse dello stesso documento.

> È possibile per esempio preparare fogli stile diversi per ottenere l'output sullo schermo, per inviare i dati a un programma per la sintesi vocale, per ottenere diverse versioni linguistiche a partire da uno stesso documento multilingua, o per nascondere o mostrare parti del documento (per esempio in base al livello dell'utente in un'applicazione per la didattica).

Il progetto XML prevede l'utilizzo di fogli stile **CSS** (Cascading Style Sheets) e **XSL** (eXtensible Style Language) per la presentazione del documento.

Per visualizzare o elaborare un documento XML serve un parser XML.

> Il **parser** XML è un'applicazione che analizza il documento (parsing), esegue la convalida rispetto alla DTD, se presente, e rende disponibili per l'elaborazione le informazioni che compongono il documento.

Applicazioni XML

Un linguaggio di markup che rispetti le specifiche XML viene definito **applicazione XML**.

Alcune note applicazioni XML sono:

XHTML definizione del linguaggio HTML come applicazione XML invece che come applicazione SGML;
XSL linguaggio per la definizione di fogli stile;
XLink linguaggio per la definizione dei collegamenti ipertestuali;
MathML (Mathematical Markup Language) linguaggio per la descrizione di espressioni matematiche;
SVG (Scalable Vector Graphics) linguaggio per la descrizione di immagini vettoriali;
X3D (eXtensible 3D) linguaggio per la descrizione di immagini tridimensionali;
SMIL (Synchronized Multimedia Integration Language) linguaggio per la descrizione di documenti multimediali che permette di specificare quando e come gli elementi che lo compongono devono apparire nel documento;
WML (Wireless Markup Language) linguaggio per sviluppare pagine Web da visualizzare su dispositivi accessibili via WAP (Wireless Application Protocol);
SOAP (Simple Object Application Protocol) protocollo di comunicazione tra applicazioni distribuite.

11.3 DOCUMENTI XML

Un documento XML è un documento di **testo** con estensione ***.xml***.

Per **scrivere** un documento XML si può usare un semplice editor di testo.
Per **visualizzare** un documento XML in un browser serve un browser che supporti XML, cioè che sia dotato di un parser XML.

Loescher Editore - Vietata la vendita e la diffusione

Un documento XML visualizzato in un **browser** viene trasformato in un documento **HTML**.

Internet Explorer per default visualizza i documenti XML con un foglio stile che presenta i dati in nero racchiusi tra i tag in rosso, con le parentesi angolari blu. Ogni elemento si può comprimere ed espandere facendo clic sul simbolo -, presente quando l'elemento è espanso, o +, presente quando l'elemento è compresso (il puntatore su questi simboli assume la forma di una mano come sui collegamenti ipertestuali).

Figura 11.1
Visualizzazione di un documento XML in Internet Explorer

Tutti i documenti XML devono essere ben formati.

Un documento XML è **ben formato** se rispetta le seguenti regole sintattiche:

- tutti i tag devono essere chiusi (*<tag> ... </tag>*); i tag vuoti (senza contenuto) possono essere chiusi inserendo la barra alla fine dell'elemento stesso (*<tag .../>*);
- i tag devono essere nidificati correttamente (i tag aperti per primi devono essere chiusi per ultimi);
- i valori degli attributi devono essere racchiusi tra virgolette.

Il parser controlla che il documento sia ben formato mentre lo interpreta; appena riscontra un errore lo segnala. Se ci sono errori il documento non viene visualizzato.

L'XML è **case sensitive**, cioè è importante la differenza tra minuscole e maiuscole.
I documenti XML possono contenere **caratteri Unicode** nella forma *&#n*; dove *n* è il codice Unicode.

Per esempio le vocali accentate possono essere inserite con:

à à	**è** è	**é** é	**ì** ì	**ò** ò	**ù** ù

Si può usare anche la codifica UTF-8:

à Alt 199 Alt 255	**è** Alt 199 Alt 249	**é** Alt 199 Alt 184
ì Alt 199 Alt 170	**ò** Alt 199 Alt 253	**ù** Alt 199 Alt 251

I **nomi dei tag** possono iniziare con una lettera, un trattino di sottolineatura o il simbolo due punti; gli altri caratteri possono essere: lettere, cifre, trattini di sottolineatura, trattini, punti e due punti; non si possono usare spazi.

Un documento XML inizia con un **prologo** che contiene almeno la dichiarazione xml.
La **dichiarazione xml** identifica la versione della specifica XML a cui è conforme il documento.

```
<?xml version="1.0"?>
```

Il prologo può contenere anche commenti, dichiarazioni di tipo di documento e istruzioni di elaborazione.

Loescher Editore - Vietata la vendita e la diffusione

I **commenti** hanno la stessa forma dei commenti HTML, cioè sono racchiusi tra <!-- e -->.
Non devono precedere la dichiarazione xml e non si possono inserire all'interno di altri elementi.
Si possono usare per descrivere il significato degli elementi o per rimuovere parti del documento.
Le **dichiarazioni di tipo di documento** specificano una o più DTD da utilizzare.
Le **istruzioni di elaborazione** sono direttive per il parser; sono racchiuse tra <? e ?>.
Un esempio di istruzione di elaborazione è l'elemento <?*xml-stylesheet* ...?> che specifica il foglio stile da applicare.

Dopo il prologo c'è l'**elemento radice**, la parte fondamentale del documento con il contenuto vero e proprio. Ogni documento ha una radice che contiene tutti gli altri elementi.

Esempio 11.1

Documento XML.

Nota

Si può vedere come appare il documento nel browser nella figura 11.1.

```
<?xml version="1.0" ?>
<libri>
    <libro isbn="978-88-8433-020-8">
        <titolo>Internet e il WWW</titolo>
        <autore>F. Scorzoni</autore>
        <edizione>
            <editore>Loescher</editore>
            <anno>2008</anno>
        </edizione>
        <prezzo>18.00</prezzo>
    </libro>
    <libro isbn="978-88-8433-025-3">
        <titolo>Il computer e la programmazione in Visual Basic</titolo>
        <autore>F. Scorzoni</autore>
        <autore>G. Costa</autore>
        <edizione>
            <editore>Loescher</editore>
            <anno>2008</anno>
        </edizione>
    <prezzo>19.00</prezzo>
    </libro>
    <libro isbn="978-88-8433-030-7">
        <titolo>Programmazione in Java</titolo>
        <autore>F. Scorzoni</autore>
        <edizione>
            <editore>Loescher</editore>
            <anno>2009</anno>
        </edizione>
        <prezzo>20.00</prezzo>
    </libro>
</libri>
```

I documenti sono costituiti da markup e da dati di tipo carattere.
I **markup** conferiscono la struttura al documento (comprendono i tag, ma anche i commenti, le dichiarazioni di tipo documento, le istruzioni di elaborazione ecc.).
I **dati di tipo carattere** sono compresi tra i tag.

Loescher Editore - Vietata la vendita e la diffusione

Nei dati di carattere non si possono usare i caratteri &, <, >, ' e " perché sono riservati al linguaggio XML. Per usare questi caratteri nel contenuto o negli attributi degli elementi bisogna usare i **riferimenti a entità**.

< <	> >	& &	' '	" "

Gli **attributi** sono coppie *nome="valore"*.
I nomi degli attributi seguono le stesse regole dei nomi di tag. Il valore dell'attributo deve essere racchiuso tra virgolette; può essere racchiuso tra apici se le virgolette fanno parte del valore.
Il valore dell'attributo non può mai contenere il carattere <; si deve sempre usare l'entità *<*.

Se si deve inserire del testo che contiene spazi e caratteri riservati come < o > si può usare una **sezione CDATA**.

```
<![CDATA[ testo ]]>
```

Il testo delle sezioni CDATA non viene elaborato dal parser.

Spazi dei nomi

Dato che i nomi degli elementi sono arbitrari potrebbero esserci in applicazioni diverse elementi con lo stesso nome e significato diverso. Per creare nomi univoci si usano gli spazi dei nomi (**namespace**).

Uno **spazio dei nomi** è una raccolta di nomi identificata da un **URI**; un URI è un identificatore univoco (un URL o un URN).

Nota

L'URI dello spazio dei nomi non richiede il collegamento a Internet per individuare la risorsa; serve solo a rendere univoci i nomi; il parser può essere predisposto per riconoscere alcuni spazi dei nomi e per elaborare opportunamente gli elementi di questi spazi.

Gli spazi dei nomi permettono di creare nomi univoci identificando i nomi con una risorsa esterna univoca. Si dice che i nomi sono **qualificati** e sono formati dal nome dello spazio dei nomi e da una parte locale.

Uno spazio dei nomi va dichiarato in un elemento dello spazio dei nomi con l'attributo ***xmlns*** e risulta disponibile nell'elemento in cui è dichiarato e in tutti i suoi elementi secondari (se dichiarato nell'elemento del documento può essere usato nell'intero documento).
Di solito nella dichiarazione dello spazio dei nomi si specifica un **prefisso**; i nomi dei prefissi seguono le stesse regole dei nomi dei tag ma non sono consentiti i due punti.

```
<pr:elemento xmlns:pr="URI">
</pr:elemento>
```

Il prefisso va specificato per tutti gli elementi che appartengono allo spazio dei nomi e va associato alla parte locale con il simbolo :.

```
<pr:elem>...</pr:elem>
```

Uno spazio dei nomi dichiarato senza specificare un prefisso diventa lo spazio dei nomi **predefinito** nell'elemento in cui è dichiarato; i nomi cioè possono essere usati senza qualificarli.

```
<elemento xmlns="URI">
    <!-- elementi dello spazio dei nomi con nomi non qualificati -->
    <elem>...</elem>
</elemento>
```

Loescher Editore - Vietata la vendita e la diffusione

Un documento XHTML specifica lo spazio dei nomi predefinito:
<html xmlns="http://www.w3.org/1999/xhtml">

Tecnologie che usano gli spazi dei nomi sono per esempio XSL e XLink.

11.4 LINK XML

Il linguaggio XML non prevede collegamenti ipertestuali. Si possono definire collegamenti ipertestuali usando il linguaggio **XLink** (un vocabolario XML).

Qualsiasi elemento può essere usato come collegamento ipertestuale. Il nome dell'elemento di collegamento non è predefinito come in HTML.

Per usare XLink bisogna fare riferimento allo spazio dei nomi *http://www.w3.org/1999/xlink*.
XLink è un vocabolario che consiste solo di **attributi**.
L'attributo ***type*** indica il tipo di link; i tipi principali sono *simple* (collegamenti semplici) e *extended* (collegamenti estesi).
In base al tipo di link sono definiti gli altri attributi che si possono usare e i valori ammessi.

Purtroppo *Firefox* supporta solo i collegamenti di tipo semplice mentre *Internet Explorer 7.0* nemmeno quelli.

I collegamenti **semplici** (*type="simple"*) permettono di definire collegamenti simili a quelli che si definiscono in HTML col tag *<a>*.

Gli attributi più importanti per un collegamento di tipo *simple* sono:

href indica dove si trova la risorsa di destinazione;
show controlla la presentazione della risorsa; i valori possibili sono: *replace* (la risorsa sostituisce il documento corrente), *new* (la risorsa viene visualizzata in una nuova finestra, come con l'attributo *target* di HTML) e *embed* (la risorsa viene inserita all'interno del documento corrente, come con il tag *<img>* di HTML);
actuate controlla il tipo di evento che attiva il link; i valori possibili sono *onRequest* (indica che il collegamento deve essere attivato da un meccanismo esterno come il clic del mouse) e *onLoad* (indica che il collegamento viene attivato automaticamente).

Per creare un collegamento bisogna aggiungere gli attributi ad un elemento XML.
Per farlo apparire come un link si possono usare i fogli stile.
In *Internet Explorer* si può usare l'evento *onclick* per definire l'azione da eseguire.

Esempio 11.2

Collegamento ipertestuale semplice che funziona come un tag HTML *<a>*.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="link.css"?>
<pagina>
<testo>questo &#232; un collegamento ipertestuale:
</testo>
<collegamento
xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple"
xlink:href="pag1.html"
onclick="location.href='pag1.html'">
```

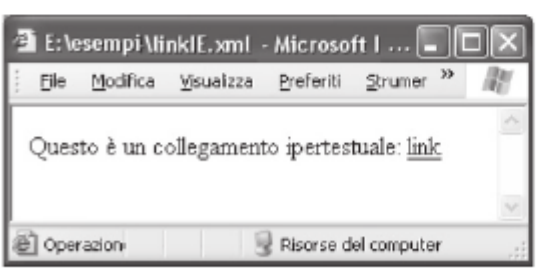


Figura 11.2 *Un collegamento ipertestuale semplice*

Loescher Editore - Vietata la vendita e la diffusione

```
link
</collegamento>
</pagina>
```

file link.css

```
collegamento {color: #0000FF; text-decoration: underline; cursor: pointer}
```

I collegamenti **estesi** permettono di definire collegamenti con destinazioni multiple e funzionanti in entrambe le direzioni.
Inoltre si possono definire i collegamenti esternamente al documento che rappresenta la sorgente del link; cioè i collegamenti possono essere contenuti in un documento distinto.

XPOINTER

Le risorse negli attributi *href* possono essere individuate da un **URI reference** cioè una combinazione di un URI e di una stringa dopo il carattere # (crosshatch), chiamata identificatore di frammento.
Gli **identificatori di frammento** in XML sono definiti dalla specifica XPointer.

```
href="uri#XPointer"
```

Nota In HTML gli identificatori di frammento sono le ancore *<a name>*.

XPointer (XML Pointer) è un linguaggio che permette di indirizzare qualsiasi cosa all'interno di un documento XML senza dover definire alcuna ancora.

Un elemento XPointer contiene una serie di termini di localizzazione che specifica una posizione nella struttura ad albero del documento.

È possibile per esempio individuare l'elemento che ha un certo *id*, l'*n*-esimo figlio di un nodo, un elemento che ha un certo tag e uno specifico valore di attributo, tutti gli elementi compresi tra due nodi individuati in un modo qualsiasi dei precedenti e così via.

11.5 DTD

La **DTD** permette di definire gli elementi che si possono usare in un documento. Si dice anche che una DTD definisce una applicazione XML, cioè un vocabolario XML o una classe di documenti.

Nota Se si creano delle applicazioni che elaborino un particolare tipo di documento XML, definendo la DTD è possibile specificare come deve essere fatto il documento per poter essere elaborato. La DTD specifica il vocabolario XML (o la classe di documenti) per cui le applicazioni sono realizzate; in tal senso definisce l'applicazione.

Se un documento utilizza una DTD può usare solo i tag definiti nella DTD e solo nel modo definito dalla DTD.
Il parser, oltre a controllare se il documento è ben formato, controlla se il documento è **valido**, cioè se è conforme alla DTD.

Nota Un parser che controlla se il documento è valido rispetto alla DTD viene detto **parser validante**; quando si utilizza un parser si può impostare in modo che sia validante e quindi richieda necessariamente la DTD per il documento o che non sia validante e quindi ignori la DTD anche se presente.

Loescher Editore - Vietata la vendita e la diffusione

La DTD può essere definita internamente o esternamente al documento XML.

La DTD **interna** viene definita con l'elemento:

```
<!DOCTYPE nomeRadice [
definizione della DTD
]>
```

La DTD **esterna** viene definita in un file con estensione ***.dtd***.
Nel documento XML si fa riferimento alla DTD esterna con l'elemento:

```
<!DOCTYPE nomeRadice SYSTEM "nomeFile.dtd">
```

Esempio 11.3

DTD esterna.

```
<?xml version="1.0" ?>
<!DOCTYPE libri SYSTEM "dtd2.dtd">
<libri>
    <libro isbn="978-88-8433-020-8">
        <titolo>Internet e il WWW</titolo>
        <autore>F. Scorzoni</autore>
        <edizione>
            <editore>Loescher</editore>
            <anno>2008</anno>
        </edizione>
        <prezzo>18.00</prezzo>
    </libro>
</libri>
```

file dtd2.dtd

```
<!ELEMENT libri (libro*)>
<!ATTLIST libro
    isbn CDATA #IMPLIED
    lingua (italiano|inglese) "italiano">
<!ELEMENT libro (titolo, autore+, edizione?, prezzo?)>
<!ELEMENT titolo (#PCDATA)>
<!ELEMENT autore (#PCDATA)>
<!ELEMENT prezzo (#PCDATA)>
<!ELEMENT edizione (editore, anno?)>
<!ELEMENT editore (#PCDATA)>
<!ELEMENT anno (#PCDATA)>
```

Validazione di un documento

Si può **validare** un documento, cioè controllare se rispetta le regole della DTD specificata, analizzandolo con un parser.
Il **parser *msxml*** si può richiamare dalla riga di comando con

```
msxml nomeFile.xml
```

I documenti XHTML si possono validare con il **Validation Service** del W3C (*http://validator.w3.org*).

Loescher Editore - Vietata la vendita e la diffusione

11.6 FOGLI STILE XSL

Per visualizzare un documento XML si possono usare i fogli stile XSL.

Un **foglio stile** è un documento XML ben formato con estensione ***.xsl***.

Il **nodo radice** deve essere ***stylesheet***. Nell'elemento *stylesheet* viene usato l'attributo *xmlns* per fare riferimento allo spazio dei nomi del linguaggio XSL.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Il documento XML fa riferimento a un foglio stile con l'istruzione di elaborazione <?***xml-stylesheet***?>.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="file.xsl"?>
```

Il linguaggio **XSL** (eXtensible Style Language), o meglio **XSLT** (XSL Transformations) è un linguaggio di **trasformazione** che permette al parser di trasformare la struttura di un documento XML in un'altra struttura. I fogli stile XSL indicano come attuare la trasformazione.

Il tipo di documento di output può essere specificato con l'attributo ***method*** del tag *output*; i valori possibili sono *xml*, *html* e *text*.

```
<xsl:output method="text" />
```

Text rappresenta il testo puro; l'output è il testo della struttura ad albero del documento.
Per specificare una trasformazione in un documento HTML basta fare in modo che il documento di output abbia come radice l'elemento <*html*>.
Il default è *xml*; il documento comincia con una dichiarazione <?*xml*?> e i riferimenti alle entità non vengono sostituiti.

La trasformazione più frequente è in un documento HTML, per permettere la visualizzazione in un browser.

Le regole di trasformazione sono specificate da tag XSL.
Tutte le regole di trasformazione devono trovarsi dentro all'elemento *xsl:stylesheet*.
Il parser elabora gli elementi XSL e riproduce il resto del testo così com'è.

XSL

Un foglio stile comprende uno o più **modelli** (**template**).

I **template** sono le regole di trasformazione; specificano la struttura del documento che deve essere generato. Ogni template specifica mediante un **pattern** o **query** gli elementi del documento XML di origine a cui applicare la trasformazione.

Le query sono di tipo **dichiarativo** (non procedurale); specificano cosa deve essere cercato nel documento ma non come deve essere eseguita la ricerca.

I pattern più semplici sono costituiti da:

- il nome di un **elemento**;
- il nome di un **attributo**, preceduto dal simbolo @.

Pattern più complicati possono essere specificati usando XPath.

I template si descrivono con l'elemento ***xsl:template***; l'attributo ***match*** specifica il pattern; tutto ciò che è all'interno dell'elemento *template* costituisce la regola di trasformazione.

Loescher Editore - Vietata la vendita e la diffusione

```
<xsl:template match="nomeNodo">
...
</xsl:template>
```

L'istruzione ***xsl:apply-templates***

```
<xsl:apply-templates />
```

richiede di applicare tutti i template disponibili per tutti i figli del nodo corrente.

L'attributo ***select***

```
<xsl:apply-templates select="nomeNodo">
```

richiede di applicare il template con il pattern corrispondente (quello con l'attributo *match* con valore *nomeNodo*).

La **trasformazione** avviene esaminando l'albero XML del documento di origine a partire dalla radice; ogni sottoalbero viene sostituito da un sottoalbero costruito eseguendo il template il cui pattern corrisponde alla radice del sottoalbero. Il procedimento è ricorsivo. Gli elementi del documento di origine a cui non si applica nessun template vengono ignorati.
La sostituzione può essere costituita da elementi letterali oppure dal risultato di istruzioni XSL.

Nota

Si possono avere fogli stile che danno risultati completamente diversi a partire dallo stesso documento.

L'istruzione ***xsl:sort*** permette di ordinare gli elementi in base al valore di un pattern (elemento o attributo) specificato con l'attributo ***select***.

```
<xsl:sort select="pattern" />
```

L'attributo ***order*** permette di specificare l'ordinamento ascendente (*ascending*) o discendente (*descending*).
L'attributo ***data-type*** permette di specificare se l'ordinamento deve essere eseguito considerando i dati di tipo numerico (*number*) o testuale (*text*).

L'istruzione *xsl:sort* può essere inserita come contenuto dell'istruzione *xsl:apply-templates* o dell'istruzione *xsl:for-each*.

Per recuperare i dati dal documento di origine si usa l'elemento ***xsl:value-of*** con l'attributo ***select***.

```
<xsl:value-of select="pattern"/>
```

Questa istruzione restituisce solo il primo elemento corrispondente al pattern.

Esempio 11.4

Visualizzazione dei dati del file *libri.xml* dell'esempio 11.1 in una tabella, in ordine discendente di *isbn*, usando modelli annidati.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="libri">
    <html>
    <head><title>XSL</title></head>
    <body>
    <h1>Tabella</h1>
    <table border="1">
    <tr>
```

Loescher Editore - Vietata la vendita e la diffusione

```
        <th>Titolo</th>
        <th>Autore</th>
        <th>Editore</th>
        <th>Anno edizione</th>
        <th>Prezzo</th>
        <th>ISBN</th>
        </tr>
        <xsl:apply-templates>
        <xsl:sort select="@isbn" order="descending"/>
        </xsl:apply-templates>
        </table>
        </body>
        </html>
</xsl:template>
<xsl:template match="libro">
        <tr>
        <td><xsl:value-of select="titolo"/> </td>
        <td><xsl:value-of select="autore"/> </td>
        <xsl:apply-templates select="edizione" />
        <td><xsl:value-of select="prezzo"/> </td>
        <td><xsl:value-of select="@isbn"/> </td>
        </tr>
</xsl:template>
<xsl:template match="edizione">
        <td><xsl:value-of select="editore"/> </td>
        <td><xsl:value-of select="anno"/> </td>
</xsl:template>
</xsl:stylesheet>
```

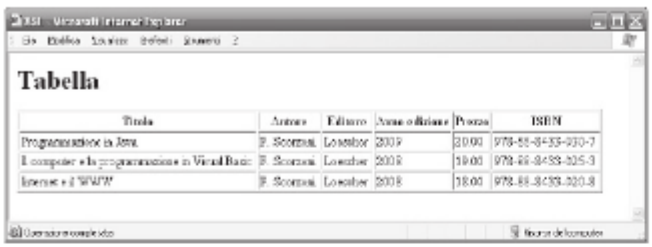


Figura 11.3 *Applicazione del foglio stile dell'esempio*

Nota

Il codice Unicode   inserisce uno spazio (spazio unificatore).

Attributi

L'istruzione *xsl:value-of* non può essere usata nel valore di un attributo.
Per inserire i dati di un elemento del documento di origine nel valore di un attributo basta indicare il nome dell'elemento tra parentesi graffe.

Esempio 11.5

Dato il file *links.xml* si può definire un foglio stile per creare un elenco di collegamenti ipertestuali usando l'elemento destinazione come valore dell'attributo *href* del tag <*a*>.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="links">
        <html>
        <head><title>XSL</title></head>
        <body>
        <h1>Collegamenti</h1>
        <ul>
        <xsl:apply-templates />
        </ul>
```

Figura 11.4 *Esecuzione dell'esempio*

Loescher Editore - Vietata la vendita e la diffusione

```
        </body>
        </html>
    </xsl:template>
    <xsl:template match="link">
        <li><a href="{destinazione}"><xsl:value-of select="descrizione" /></a></li>
    </xsl:template>
    </xsl:stylesheet>
```

file links.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="links1.xsl"?>
<links>
<link><descrizione>Prima pagina</descrizione><destinazione>pag1.html</destinazione></link>
<link><descrizione>Seconda pagina</descrizione><destinazione>pag2.html</destinazione></link>
</links>
```

Si può eseguire un ciclo con l'istruzione ***xsl:for-each***.

```
<xsl:for-each select="pattern">
...
</xsl:for-each>
```

Nota

Questa istruzione scandisce i figli di un elemento ma non visualizza nulla, per visualizzare dei dati bisogna inserire all'interno del ciclo l'istruzione *xsl:value-of*.

Esempio 11.6

Visualizzazione dei dati del file *libri.xml* in una tabella, in ordine discendente di *isbn*, usando l'istruzione *xsl:for-each*. L'output è lo stesso dell'esempio 11.4.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="libri">
    <html>
    <head><title>XSL</title></head>
    <body>
    <h1>Tabella</h1>
    <table border="1">
    <tr>
    <th>Titolo</th>
    <th>Autore</th>
    <th>Editore</th>
    <th>Anno edizione</th>
    <th>Prezzo</th>
    <th>ISBN</th>
    </tr>
    <xsl:for-each select="libro">
    <xsl:sort select="@isbn" order="descending"/>
        <tr>
        <td><xsl:value-of select="titolo"/> </td>
        <td><xsl:value-of select="autore"/> </td>
        <xsl:apply-templates select="edizione" />
```

Figura 11.5 *Applicazione del foglio stile dell'esempio*

Loescher Editore - Vietata la vendita e la diffusione

```
            <td><xsl:value-of select="prezzo"/> </td>
            <td><xsl:value-of select="@isbn"/> </td>
            </tr>
        </xsl:for-each>
        </table>
        </body>
        </html>
    </xsl:template>
    <xsl:template match="edizione">
        <td><xsl:value-of select="editore"/> </td>
        <td><xsl:value-of select="anno"/> </td>
    </xsl:template>
    </xsl:stylesheet>
```

Si può realizzare una struttura condizionale con l'istruzione ***xsl:if***. L'espressione logica che costituisce la condizione è specificata nell'attributo ***test***.

```
<xsl:if test="condizione">
...
</xsl:if>
```

Nota

L'istruzione *xsl:if* permette di realizzare soltanto strutture condizionali con un solo ramo; per realizzare strutture condizionali con due o rami (tipo *if ... then ... else*) bisogna usare l'istruzione *xsl:choose*.

L'istruzione ***xsl:choose*** permette di realizzare strutture condizionali a due o più rami. Si possono testare diverse condizioni ed eseguire istruzioni di conseguenza inserendo nell'elemento *xsl:choose* una istruzione ***xsl:when*** per ogni possibile condizione. L'istruzione ***xsl:otherwise*** permette di specificare cosa fare quando nessuna delle condizioni specificate è verificata.

```
<xsl:choose>
<xsl: when test="pattern">...</xsl:when>
<xsl: when test="pattern">...</xsl:when>
...
<xsl:otherwise>...</xsl:otherwise>
</xsl:choose>
```

XPATH

I pattern possono essere specificati usando XPath.

XPath è un linguaggio che permette di indirizzare parti di un documento XML.

In XPath un documento XML viene considerato come una struttura ad **albero** in cui ogni parte del documento è un **nodo**. Sono nodi per esempio gli elementi, gli attributi e il testo degli elementi. C'è un solo **nodo radice**; tutti gli altri nodi hanno un genitore (**parent**) e possono avere dei figli (**child**). Ogni elemento ha come figli gli elementi che contiene; i nodi di testo sono figli degli elementi che li contengono. Gli attributi hanno un parent (il nodo che li contiene) ma non sono figli di quel nodo.

Un'espressione XPath può dare come risultato un insieme di nodi oppure un valore booleano, numerico o stringa. La valutazione di una espressione avviene in un contesto.

Loescher Editore - Vietata la vendita e la diffusione

Il contesto nell'albero che rappresenta il documento indica il nodo che si considera corrente (**nodo di contesto**).
Le istruzioni *xsl:template* e *xsl:for-each* cambiano il contesto in base al pattern specificato.

Nei pattern XPath si possono utilizzare:

i **simboli**:

.	rappresenta l'elemento corrente, o nodo di contesto;
..	rappresenta l'elemento genitore, cioè il nodo parent del nodo di contesto;
@	permette di accedere agli attributi del nodo di contesto;
|	indica delle alternative;
/	scende di un livello nell'albero;
//	scende di un numero qualsiasi di livelli nell'albero;

e i **test dei nodi**:

nome	è *true* per ogni nodo che ha quel nome;
*	è *true* per ogni elemento figlio del nodo di contesto;
node()	è *true* per ogni nodo;
text()	è *true* per ogni nodo di testo.

In particolare:

@*nome*	seleziona l'attributo specificato del nodo di contesto;
@*	seleziona ogni attributo del nodo di contesto.

libro/titolo
seleziona l'elemento *titolo* figlio dell'elemento *libro*;

//titolo
seleziona l'elemento *titolo* a qualsiasi livello nella gerarchia degli elementi;

./*
seleziona il primo figlio del nodo di contesto;

autore | titolo
seleziona l'elemento *autore* o l'elemento *titolo*, figli del nodo di contesto;

@isbn
seleziona l'attributo *isbn* del nodo di contesto.

Esempio 11.7

Visualizzazione dei dati del file *libri.xml* in una tabella, in ordine discendente di *isbn*, indicando anche più autori per ogni libro, usando XPath.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="libri">
    <html>
    <head><title>XSL</title></head>
    <body>
    <h1>Tabella</h1>
    <table border="1">
    <tr><th>Titolo</th>
    <th>Autore</th>
    <th>Editore</th>
```

Figura 11.6 *Applicazione del foglio stile dell'esempio*

Loescher Editore - Vietata la vendita e la diffusione

```
        <th>Anno edizione</th>
        <th>Prezzo</th>
        <th>ISBN</th>
        </tr>
        <xsl:for-each select="libro">
        <xsl:sort select="*/anno" order="descending"/>
            <tr>
            <td><xsl:value-of select="titolo"/> </td>
            <td>
            <xsl:for-each select="autore">
                <xsl:value-of select="."/> <br/>
            </xsl:for-each>
            </td>
            <td><xsl:value-of select="edizione/editore"/> </td>
            <td><xsl:value-of select="edizione/anno"/> </td>
            <td><xsl:value-of select="prezzo"/> </td>
            <td><xsl:value-of select="@isbn"/> </td>
            </tr>
        </xsl:for-each>
        </table>
        </body>
        </html>
    </xsl:template>
    </xsl:stylesheet>
```

Le sottoquery permettono di fare operazioni di selezione (o filtro) su un insieme di nodi.

Le espressioni di selezione vanno inserite tra parentesi quadre.

[ ] operatori di sottoquery

[nome="valore"]
seleziona gli elementi in cui il figlio specificato ha il valore richiesto;

[nome]
seleziona gli elementi in cui è presente il figlio specificato;

[@attributo="valore"]
seleziona gli elementi in cui l'attributo specificato ha il valore richiesto;

[@attributo]
seleziona gli elementi in cui è presente l'attributo specificato.

Nota

Si possono anche inserire più condizioni a diversi livelli del pattern.

Esempio 11.8

Visualizzazione dei dati dei libri con anno di edizione 2009.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="libri">
    <html>
    <head><title>XSL</title></head>
    <body>
```

```
        <xsl:for-each select="libro[edizione/anno='2009']">
            <xsl:value-of select="titolo"/><xsl:text> - </xsl:text>
            <xsl:value-of select="autore"/>
            <br/>
        </xsl:for-each>
        </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Figura 11.7 *Applicazione del foglio stile dell'esempio*

Nota

L'elemento *xsl:text* permette di inserire testo nel documento di output; in questo caso inserisce uno spazio, un trattino e un altro spazio. Per inserire uno spazio si possono usare anche i codici Unicode   o   (spazio unificatore).

VERIFICA

QUESTIONARIO

1. Che cos'è un linguaggio di markup?
2. Che cos'è SGML?
3. Che cos'è XML e quali sono le principali differenze rispetto a SGML e HTML?
4. Quali sono alcune applicazioni XML?
5. Come si può scrivere e visualizzare un documento XML?
6. Che cosa si intende per documento XML ben formato?
7. A che cosa servono gli spazi dei nomi?
8. Come si possono inserire collegamenti ipertestuali in un documento XML?
9. Che cos'è XLink?
10. Che cos'è XPointer?
11. Che cos'è la DTD?
12. Che cosa si intende per validazione di un documento?
13. Qual è la differenza tra DTD esterna e interna?
14. Che cos'è XSL?
15. Che cosa fa un foglio stile XSLT?
16. Quali tipi di documenti si possono ottenere applicando un foglio stile XSLT a un documento XML?
17. Come si applica un foglio stile a un documento XML?
18. Che cosa sono template e pattern?
19. A che cosa servono i tag *xsl:template* e *xsl:apply-templates*?
20. A che cosa serve e dove si può usare il tag *xsl:sort*?
21. A che cosa serve il tag *xsl:value-of*?
22. Come si possono inserire i dati di un elemento nel valore di un attributo?
23. Come si può realizzare un ciclo?
24. Come si possono eseguire delle istruzioni solo se sono verificate delle condizioni?
25. Che cos'è XPath?
26. Come si possono eseguire operazioni di selezione su un insieme di nodi?

Loescher Editore - Vietata la vendita e la diffusione

Esercizi

1 ..... Scrivere un documento XML con la seguente struttura:

```
<?xml version="1.0" ?>
<anagrafica>
    <persona professione="" annoNascita="">
        <nominativo>
            <cognome>...</cognome>
            <nome>...</nome>
        </nominativo>
        <indirizzo>
            <via>...</via>
            <citta>...</citta>
        </indirizzo>
    </persona>
    ... altri elementi persona
</anagrafica>
```

poi scrivere i fogli stile per:

1) visualizzare i dati, una persona per riga, usando modelli annidati;
2) visualizzare i dati in una tabella usando modelli annidati;
3) visualizzare i dati in una tabella usando l'istruzione *for-each*;
4) visualizzare i dati degli studenti (*professione="studente"*);
5) visualizzare i dati evidenziando le persone per cui non è stato specificato l'attributo *professione*;
6) visualizzare i dati, una persona per riga, usando XPath;
7) visualizzare i dati in una tabella usando XPath;
8) visualizzare i dati delle persone che abitano in una città specificata.

PROPOSTE DI LAVORO

1 ..... Scrivere un file XML per descrivere:

1) i dati anagrafici di alcune persone;
2) una rubrica telefonica (con più numeri di telefono per esempio casa, lavoro, cellulare e indirizzo e-mail);
3) un catalogo di prodotti di una azienda;
4) un elenco di domande e risposte (una sola risposta per ogni domanda);
5) un elenco di articoli con titolo, autore, data, sommario e testo di ciascuno;
6) il menu di una pizzeria con i nomi delle pizze, gli ingredienti e il prezzo;
7) un catalogo di CD con titolo, autore, data, e lista dei brani contenuti con la durata;
8) un messaggio e-mail con mittente, destinatario, cc, bcc, elenco allegati, oggetto e testo;
9) una fattura con intestazione venditore, cliente e vettore, data, descrizione dei prodotti con quantità e prezzo, totale e IVA;
10) i dati di una o più classi di studenti;
11) una agenda di appuntamenti;
12) il catalogo dei libri di una biblioteca con la registrazione dei prestiti;
13) un articolo di una rivista;
14) un libro;
15) un elenco degli studenti di una classe con i voti riportati per ogni materia;

Loescher Editore - Vietata la vendita e la diffusione

16) un elenco di offerte di viaggio con destinazione, durata, prezzo ed elenco delle date di partenza;
17) un poligono;
18) un elenco di figure geometriche (prevedendo le informazioni necessarie per disegnarle);
19) un elenco di domande e risposte per un test del tipo vero/falso;
20) un elenco di domande e risposte per un test a risposta multipla (domande con varie risposte per ogni domanda e identificazione della risposta esatta).

2 ..... Scrivere un foglio stile per visualizzare opportunamente un articolo di una rivista descritto da un file del tipo:

```
<articolo>
    <titolo></titolo>
    <autore></autore>
    <data></data>
    <testo></testo>
</articolo>
```

3 ..... Scrivere un foglio stile per visualizzare opportunamente un messaggio di posta descritto da un file del tipo:

```
<email>
    <mittente></mittente>
    <destinatario></destinatario>
    <cc></cc>
    ... altri elementi cc
    <bcc></bcc>
    ... altri elementi bcc
    <allegato></allegato>
    ... altri elementi allegato
    <oggetto></oggetto>
    <testo></testo>
</email>
```

4 ..... Dato un documento contenente dei messaggi del tipo:

```
<messaggi>
    <messaggio numero="n">
        testo del messaggio
    </messaggio>
    ... altri elementi messaggio
</messaggi>
```

- scrivere un foglio stile per visualizzare opportunamente i messaggi;
- scrivere un foglio stile per visualizzare il messaggio numero *n*.

5 ..... Dato un documento relativo a una rubrica telefonica del tipo:

```
<rubrica>
    <persona>
        <nominativo>Cognome Nome</nominativo>
        <telefono>numero telefonico</numero>
    </persona>
    ... altri elementi persona
</rubrica>
```

- scrivere un foglio stile per visualizzare opportunamente i dati;
- scrivere un foglio stile per visualizzare il numero telefonico della persona col nominativo specificato.

Loescher Editore - Vietata la vendita e la diffusione

Elaborazione di documenti XML 12

Prerequisiti

- XML e XSL
- Linguaggio Java, tecnologie JSP e ASP.NET

Obiettivi

Conoscenze

- Java e XML: API SAX e DOM di Java, uso dell'API DOM e classi del package *javax.xml.transform*
- ASP.NET e XML: controlli Xml e XmlDataSource, classi XmlTextReader, XmlTextWriter, XmlDocument e XmlDataDocument

Abilità/capacità

- Realizzare applicazioni Java e pagine JSP che elaborano documenti XML e che applicano fogli stile XSL a documenti XML
- Utilizzare i controlli e le classi specifiche per elaborare documenti XML in applicazioni server side con tecnologia ASP.NET e trasformare database in file XML e viceversa mediante DataSet

Loescher Editore - Vietata la vendita e la diffusione

12.1 JAVA E XML

API SAX E DOM

Un'applicazione accede al contenuto di un documento sfruttando le API (Application Program Interface) messe a disposizione da un **parser** XML.

Il **parser** analizza il documento XML e rende disponibili le informazioni all'applicazione.

Java supporta due **API** complementari per l'elaborazione di un documento XML:

- **SAX** (Simple API for XML);
- **DOM** (Document Object Model for XML).

SAX

L'API **SAX** (Simple API for XML) usa un procedimento **basato sugli eventi** (modello event-driven).

Il parser legge il documento XML sequenzialmente e ad ogni evento di parsing (inizio del documento, inizio di un elemento, sequenza di caratteri fra due tag, fine di un elemento, fine del documento, ecc.) richiama un particolare metodo associato a tale evento, chiamato handler o gestore dell'evento.

SAX non mantiene in memoria alcuna rappresentazione del documento e non fornisce alcun supporto per la creazione e la modifica di documenti.

DOM

Con l'API **DOM** un documento XML è visto come un insieme di elementi annidati, strutturati in modo gerarchico a formare un albero di oggetti.

L'elemento al più alto livello della gerarchia è il documento (oggetto di tipo *Document*); ogni elemento del documento è un nodo (oggetto *Node*); anche il testo contenuto in un elemento è un nodo; i nodi figli di un nodo costituiscono una *NodeList*; l'insieme degli attributi di un nodo costituisce una *NamedNodeMap*.

Tabella 12.1 *Oggetti del modello a oggetti XML*

Document	il documento, che contiene tutti gli oggetti;
Node	un nodo (a qualsiasi livello);
NodeList	un elenco di nodi di pari livello figli di un nodo;
NamedNodeMap	insieme di attributi.

Un parser basato su DOM costruisce in memoria l'albero degli elementi del documento e lo restituisce all'applicazione come oggetto di tipo *Document*.

DOM rispetto a SAX ha il vantaggio di permettere di modificare documenti preesistenti e di creane di nuovi.

ELABORAZIONE DI UN DOCUMENTO XML CON DOM

Per elaborare un documento XML con **DOM** bisogna:

- creare un oggetto ***DocumentBuilderFactory***;

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
```

Loescher Editore - Vietata la vendita e la diffusione

il metodo statico *newInstance()* restituisce un riferimento a un oggetto che rappresenta un'implementazione concreta della classe *DocumentBuilderFactory* che è astratta;

- usare l'oggetto *factory* per creare un oggetto ***DocumentBuilder***;

  DocumentBuilder domParser = dbf.newDocumentBuilder();

  il metodo *newDocumentBuilder()* di *DocumentBuilderFactory* crea un oggetto che permette di interagire con il parser; questo metodo può generare l'eccezione:

  ParserConfigurationException (javax.xml.parsers)
  se il parser non è in grado di supportare la configurazione richiesta.

- effettuare il parsing del documento con il metodo ***parse()*** di *DocumentBuilder*.

Document **parse**(File f)
Document **parse**(String uri)
carica ed elabora il documento specificato creando l'albero degli oggetti.

Partendo dall'elemento **radice** dell'albero si possono usare i metodi di ***Document*** e di ***Node*** per navigare attraverso gli elementi e per modificare il documento.

In cima alla gerarchia dei nodi di un documento si trova il nodo ***Document*** che viene restituito dal metodo *parse()*. Il nodo *Document* permette di accedere agli altri nodi.

Il punto di partenza per esplorare l'albero è l'elemento **radice** del documento, che si ottiene con il metodo ***getDocumentElement()*** di *Document* che restituisce come tipo *Element*.
Per accedere ai **figli** del nodo radice o di un qualsiasi nodo si usa il metodo ***getChildNodes()*** di *Node* che restituisce un oggetto di tipo *NodeList* che contiene i figli.
NodeList è una collezione ordinata di riferimenti a oggetti di tipo *Node* (o meglio di uno dei suoi sottotipi).
Per accedere agli **attributi** di un elemento si usa il metodo ***getAttributes()*** di *Node* che restituisce un oggetto di tipo *NamedNodeMap* che contiene gli attributi.
NamedNodeMap rappresenta una collezione di oggetti di tipo *Node*, che, dato che rappresentano attributi, sono in realtà di tipo *Attr*.

Creazione e modifica dei documenti

Si può **creare** un oggetto *Document* vuoto con il metodo ***newDocument()*** di *DocumentBuilder*.

Document documento = domParser.newDocument();

Per **creare** nuovi **nodi** si usano i metodi dell'interfaccia *Document*. Per **inserire** i nuovi nodi nel documento o per **modificare** nodi esistenti si usano i metodi dell'interfaccia *Node*.
Per aggiungere, cancellare o modificare gli **attributi** di un elemento si possono usare i metodi di *NamedNodeMap* o i metodi di *Element*.

Durante la creazione dell'albero possono essere generate eccezioni *DOMException*.

Per scrivere il documento XML in un file si può:

- creare un oggetto ***DomImplementation*** dal documento con il metodo *getImplementation()* di *Document*;

  DOMImplementation impl = documento.getImplementation();

- usare il metodo *getFeature()* di *DomImplementation* per ottenere un oggetto ***DOMImplementationLS***;

  DOMImplementationLS implLS = (DOMImplementationLS)impl.getFeature("LS", "3.0");

Loescher Editore - Vietata la vendita e la diffusione

- usare il metodo *createLSSerializer()* di *DomImplementationLS* per ottenere un oggetto ***LSSerializer***;

  LSSerializer ser = implLS.createLSSerializer();

- usare il metodo ***writeToString()*** di *LSSerializer* per ottenere il documento XML sotto forma di stringa che può essere visualizzata o scritta in un file; il documento XML così prodotto non contiene spazi o ritorni a capo;

  String output = ser.writeToString(documento);

Esempio 12.1

Creazione di un documento XML e scrittura in un file.

```
import java.io.*;
import org.w3c.dom.*;
import org.w3c.dom.ls.*;
import javax.xml.parsers.*;
import java.util.*;

public class CreaXml {
    private Document documento;

    public CreaXml() {
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        try {
            DocumentBuilder builder = dbf.newDocumentBuilder();
            documento = builder.newDocument();
        }
        catch (ParserConfigurationException e) {
            e.printStackTrace();
        }
        Element radice = documento.createElement("libri");
        documento.appendChild(radice);
        do {
            Node libro = creaLibro(documento);
            radice.appendChild(libro);
        } while (lettura("Altro libro?(S/N)").equalsIgnoreCase("s"));
        DOMImplementation impl = documento.getImplementation();
        DOMImplementationLS implLS = (DOMImplementationLS)impl.getFeature("LS", "3.0");
        LSSerializer ser = implLS.createLSSerializer();
        String output = ser.writeToString(documento);
        System.out.println(output);
        try {
            PrintStream file = new PrintStream(new FileOutputStream(lettura("Nome del file xml")));
            file.print(output);
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }

    public Node creaLibro(Document documento) {
        Element libro = documento.createElement("libro");
        Attr isbn = documento.createAttribute("isbn");
        isbn.setValue(lettura("ISBN"));
```

Loescher Editore - Vietata la vendita e la diffusione

```
        libro.setAttributeNode(isbn);
        Element titolo = documento.createElement("titolo");
        titolo.appendChild(documento.createTextNode(lettura("Titolo:")));
        libro.appendChild(titolo);
        Element autore = documento.createElement("autore");
        autore.appendChild(documento.createTextNode(lettura("Autore:")));
        libro.appendChild(autore);
        Element edizione = documento.createElement("edizione");
        Element editore = documento.createElement("editore");
        editore.appendChild(documento.createTextNode(lettura("Editore:")));
        edizione.appendChild(editore);
        Element anno = documento.createElement("anno");
        anno.appendChild(documento.createTextNode(lettura("Anno edizione:")));
        edizione.appendChild(anno);
        libro.appendChild(edizione);
        Element prezzo = documento.createElement("prezzo");
        prezzo.appendChild(documento.createTextNode(lettura("Prezzo:")));
        libro.appendChild(prezzo);
        return libro;
    }

    private String lettura(String domanda){
        Scanner tastiera = new Scanner(System.in);
        System.out.println(domanda);
        return tastiera.nextLine();
    }
}
```

```
Prompt dei comandi
>java ProvaCreaXml
ISBN
978-88-8433-020-8
Titolo:
Internet e il WWW
Autore:
F. Scorzoni
Editore:
Loescher
Anno edizione:
2008
Prezzo:
18,00
Altro libro?(S/N)
N
<?xml version="1.0" encoding="UTF-16"?>
<libri><libro isbn="978-88-8433-020-8"><titolo>Internet e il WWW</titolo><autore
>F. Scorzoni</autore><edizione><editore>Loescher</editore><anno>2008</anno></edi
zione><prezzo>18,00</prezzo></libro></libri>
Nome del file xml
libri.xml
```

Figura 12.1 *Esecuzione dell'esempio*

Applicare un foglio stile XSL a un documento XML

Per applicare un foglio stile a un documento XML servono alcune classi del package ***javax.xml.transform***; bisogna:

- creare un oggetto ***TransformerFactory***;

  TransformerFactory tf = TransformerFactory.newInstance();

  il metodo statico *newInstance()* restituisce un riferimento a un oggetto che rappresenta un'implementazione concreta della classe *TransformerFactory* che è astratta;

- usare l'oggetto *factory* per creare un oggetto ***Transformer*** specificando il foglio stile da utilizzare;

  Transformer t = tf.newTransformer(new StreamSource("file.xsl"));

Loescher Editore - Vietata la vendita e la diffusione

- applicare il foglio stile al documento XML e generare un output con il metodo ***transform()*** della classe *Transformer*.

void **transform**(Source xmlSource, Result outputTarget)
applica il foglio stile al documento XML specificato come primo argomento e produce l'output specificato come secondo argomento.

Esempio 12.2

Applicazione di un foglio stile a un documento XML con memorizzazione in un file del codice HTML generato.

```
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import java.io.*;

public class ApplicaXSL {
	public static void main(String[] args) {
		TransformerFactory tf = TransformerFactory.newInstance();
		try {
			Transformer t = tf.newTransformer(new StreamSource("libri.xsl"));
			t.transform(new StreamSource("libri.xml"), new StreamResult(new FileOutputStream("libri.html")));
		}
		catch (Exception e) {System.out.println(e);}
	}
}
```

12.2 JSP E XML

Generazione di codice XML

Una pagina JSP può generare codice XML impostando il tipo di contenuto a *text/xml* con l'attributo ***contentType*** della direttiva ***page***.

Esempio 12.3

Generazione di un documento XML.

```
<%@ page contentType="text/xml" %>
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="libri.xsl"?>
<libri>
	<libro isbn="978-88-8433-020-8">
		<titolo>Internet e il WWW</titolo>
		<autore>F. Scorzoni</autore>
		<edizione>
			<editore>Loescher</editore>
			<anno>2008</anno>
		</edizione>
		<prezzo>18.00</prezzo>
	</libro>
	<libro isbn="978-88-8433-026-0">
		<titolo>Il computer e la programmazione in Pascal</titolo>
```

Loescher Editore - Vietata la vendita e la diffusione

```
        <autore>F. Scorzoni</autore>
        <edizione>
            <editore>Loescher</editore>
            <anno>2008</anno>
        </edizione>
        <prezzo>20.00</prezzo>
    </libro>
</libri>
```

Figura 12.2 *Esecuzione dell'esempio*

Elaborazione di documenti XML

In una pagina JSP si può elaborare un documento XML con **SAX** o **DOM**.

Esempio 12.4

Visualizzazione di un documento XML con DOM.

```
<%@ page import="java.io.*, org.w3c.dom.*, javax.xml.parsers.*" %>
<%!
String s="";
public void esaminaNodo(Node nodo) {
    switch (nodo.getNodeType()) {
        case Node.ELEMENT_NODE:
            s = s + "Elemento: " + nodo.getNodeName() + "<br/>";
            NamedNodeMap attributi = nodo.getAttributes();
            if (attributi.getLength()>0) {
                s = s + "   Attributi: ";
                for (int i = 0; i < attributi.getLength(); i++){
                    Attr attributo = (Attr) attributi.item(i);
                    s = s + attributo.getNodeName() + " = " + attributo.getNodeValue() + " ";
                }
                s = s + "<br/>";
            }
            esaminaFigli(nodo.getChildNodes());
            s = s + "Fine elemento: " + nodo.getNodeName() + "<br/>";
            break;
        case Node.CDATA_SECTION_NODE:
        case Node.TEXT_NODE:
            Text testo = (Text)nodo;
            if (!testo.getNodeValue().trim().equals("")) {
                s = s + "   Testo: " + testo.getNodeValue() + "<br/>";
            }
        break;
    }
}
public void esaminaFigli(NodeList figli) {
    if (figli.getLength() > 0) {
        for (int i = 0; i < figli.getLength(); i++) {
            esaminaNodo(figli.item(i));
        }
    }
}
```

```
%>
<html>
<head><title>JSP</title></head>
<body>
<%
    DocumentBuilderFactory dbf =
        DocumentBuilderFactory.newInstance();
    DocumentBuilder builder = dbf.newDocumentBuilder();
    Document documento =
        builder.parse(application.getRealPath("/")
            + "/JSPeXML/libri.xml");
    Element radice = documento.getDocumentElement();
    esaminaNodo(radice);
    out.println(s);
%>
</body>
</html>
```

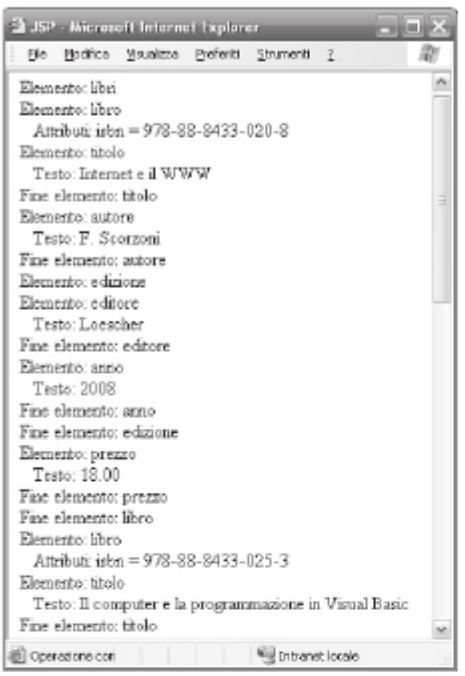


Figura 12.3 *Esecuzione dell'esempio*

Applicare un foglio stile XSL a un documento XML

Esempio 12.5

Pagina che applica un foglio stile a un documento XML e manda il risultato al browser per la visualizzazione.

```
<%@ page import="javax.xml.transform.*, javax.xml.transform.stream.*, java.io.*" %>
<html>
<head><title>JSP</title></head>
<body>
<%
TransformerFactory tf = TransformerFactory.newInstance();
Transformer t = tf.newTransformer(new StreamSource(application.getRealPath("/") + "/JSPeXML/es37/libri.xsl"));
t.transform(new StreamSource(application.getRealPath("/") + "/JSPeXML/libri.xml"), new StreamResult(out));
%>
</body>
</html>
```

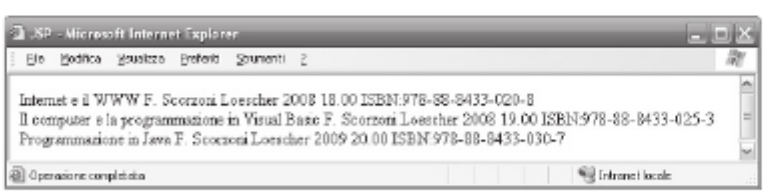


Figura 12.4 *Esecuzione dell'esempio*

12.3 ASP.NET E XML

Il controllo Xml

Per visualizzare un documento XML in una pagina ASP.NET basta utilizzare il controllo Web server ***Xml***.

Loescher Editore - Vietata la vendita e la diffusione

L'attributo ***documentSource*** specifica il file xml da visualizzare; l'attributo ***transformSource*** permette di specificare un foglio stile XSL da applicare.

Esempio 12.6

Pagina che applica un foglio stile a un documento XML e manda il risultato al browser per la visualizzazione.

```
<html>
<head>
<title>ASP.NET</title>
</head>
<body>
<form runat="server">
<asp:xml runat="server" documentSource="libri.xml" transformSource="libri.xsl" />
</form>
</body>
</html>
```

Figura 12.5 *Esecuzione dell'esempio*

Lettura e scrittura di documenti XML

Lettura

Per leggere un file XML si può usare la classe ***XmlTextReader*** del namespace *System.Xml*.

Costanti che indicano i tipi di nodi:

XmlNodeType.Text	nodo di testo
XmlNodeType.Element	elemento
XmlNodeType.Attribute	attributo

Esempio 12.7

Lettura di un file XML con *XmlTextReader*.

```
<% @Import namespace="System.Xml" %>

<script runat="server">
Sub Page_Load
Dim xmlr As XmlTextReader
xmlr = New XmlTextReader(Server.MapPath("libri.xml"))
xmlr.WhitespaceHandling = WhitespaceHandling.None
Do While xmlr.read()
If xmlr.NodeType = XmlNodeType.Text Then
Response.Write(xmlr.ReadString() & "<br/>")
End If
Loop
End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
</body>
</html>
```

Figura 12.6 *Esecuzione dell'esempio*

Loescher Editore - Vietata la vendita e la diffusione

xmlr.WhitespaceHandling = WhitespaceHandling.None dice di non considerare gli spazi vuoti.

Nota

Scrittura

Per scrivere un file XML si può usare la classe ***XmlTextWriter*** del namespace *System.Xml*. Nel costruttore dell'oggetto *XmlTextWriter* si specifica la posizione del file.

La cartella deve avere il permesso di scrittura.

Nota

La proprietà ***Formatting*** specifica il tipo di formattazione da applicare.

Per scrivere nel documento sono disponibili i metodi:

WriteStartDocument()	scrive la dichiarazione del documento;
WriteStartElement()	scrive un tag iniziale;
WriteEndElement()	scrive un tag di chiusura per un elemento;
WriteEndDocument()	scrive i tag di chiusura per tutti gli elementi rimasti aperti;
WriteElementString()	scrive il testo contenuto in un elemento;
WriteRaw()	scrive direttamente una stringa (per esempio tutto un elemento dal tag aperto al tag chiuso).

Esempio 12.8

Scrittura di un file XML con *XmlTextWriter*.

```
<% @Import namespace="System.Xml" %>

<script runat="server">
Sub Page_Load
Dim xmlw As XmlTextWriter
xmlw = New XmlTextWriter(Server.MapPath("nuovo.xml"), nothing)
xmlw.Formatting = Formatting.Indented
xmlw.WriteStartDocument()
xmlw.WriteStartElement("libri")
xmlw.WriteStartElement("libro")
xmlw.WriteElementString("titolo", "La divina commedia")
xmlw.WriteElementString("autore", "Dante")
xmlw.WriteEndElement()
xmlw.WriteEndDocument()
xmlw.Close
End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
</body>
</html>
```

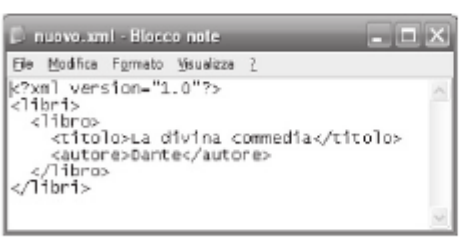


Figura 12.7 *Il file creato dall'esempio*

XmlTextReader e *XmlTextWriter* non implementano il DOM.

Nota

Loescher Editore - Vietata la vendita e la diffusione

XmlDataSource e TreeView

Ci si può collegare a un file XML con il controllo ***XmlDataSource***. L'attributo ***DataFile*** specifica il nome del file XML.

Per la visualizzazione si può utilizzare il controllo Web server ***TreeView***.
TreeView è un controllo gerarchico che permette di visualizzare il contenuto del file configurando i collegamenti ai nodi dell'albero nella sezione ***DataBindings***. Ogni elemento ***TreeNodeBinding*** indica la profondità (*Depth*) del nodo XML (*DataMember*) specificato; l'attributo *TextField="#innertext"* richiede di visualizzare nel *TreeView* il valore associato al nodo.

Esempio 12.9

Visualizzazione di un documento XML in modo gerarchico con *XmlDataSource* e *TreeView*.

```
<html>
<head>
<title>ASP.NET</title>
</head>
<body>
<form runat="server">
<asp:XmlDataSource runat="server" id="fileXml" DataFile="libri.xml" />
<asp:TreeView runat="server" DataSourceId="fileXml">
<DataBindings>
<asp:TreeNodeBinding Depth="2" DataMember="titolo" TextField="#innertext" />
<asp:TreeNodeBinding Depth="2" DataMember="autore" TextField="#innertext" />
<asp:TreeNodeBinding Depth="3" DataMember="editore" TextField="#innertext" />
<asp:TreeNodeBinding Depth="3" DataMember="anno" TextField="#innertext" />
<asp:TreeNodeBinding Depth="2" DataMember="prezzo" TextField="#innertext" />
</DataBindings>
</asp:TreeView>
</form>
</body>
</html>
```

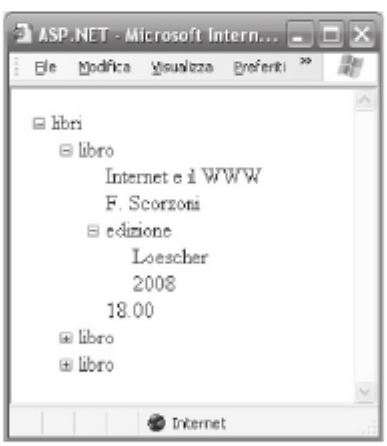


Figura 12.8 *Esecuzione dell'esempio*

DataSet e XML

Un oggetto ***DataSet*** permette di trasformare automaticamente un database in un file XML e viceversa.
I metodi principali di *DataSet* per lavorare con XML sono:

GetXML()	restituisce la rappresentazione XML dei dati del database;
WriteXML()	scrive i file XML che rappresentano i dati del database (serializzazione);
ReadXML()	popola il *DataSet* leggendo i dati dal file XML specificato.

Loescher Editore - Vietata la vendita e la diffusione

Esempio 12.10

Trasformazione di un database in un file XML mediante un *DataSet*.

```
<% @Import NameSpace="System.Data.OleDb" %>
<% @Import NameSpace="System.Data" %>

<script runat="server">
Sub Page_Load()
Dim origine As OleDbConnection
Dim selezione As OleDbDataAdapter
Dim dati As DataSet
Dim lettore As DataTableReader
origine = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
Server.MapPath("cap.mdb"))
origine.Open()
selezione = New OleDbDataAdapter("SELECT * FROM elenco", origine)
dati = new DataSet()
selezione.Fill(dati, "elenco")
origine.close()
dati.WriteXml(Server.MapPath("dati.xml"))
End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
</body>
</html>
```

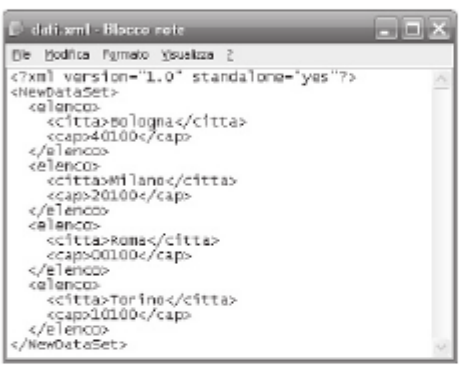


Figura 12.9 *Il file creato dall'esempio*

Esempio 12.11

Trasformazione di un file XML in un database mediante un *DataSet*.

```
<% @Import NameSpace="System.Data" %>

<script runat="server">
Sub Page_Load()
Dim dati As DataSet
dati = new DataSet("Primo")
dati.ReadXml(Server.MapPath("libri.xml"))
Dim lettore As DataTableReader
lettore = dati.CreateDataReader()
while lettore.Read()
Response.Write(lettore(0) & " " & lettore(2) & "<br />")
End While
lettore.close()
End Sub
</script>

<html>
<head><title>ASP.NET</title></head>
<body>
</body>
</html>
```

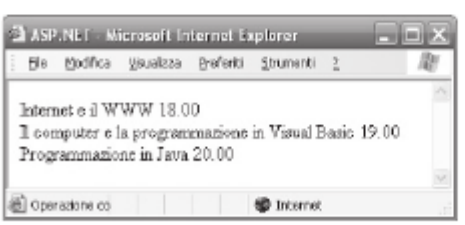


Figura 12.10 *Esecuzione dell'esempio*

Loescher Editore - Vietata la vendita e la diffusione

DOM XML

Per utilizzare il DOM si può usare la classe ***XmlDocument*** (*XmlDocument* estende la classe *XmlNode* che rappresenta un nodo).

```
Dim documento As XmlDocument
documento = new XmlDocument()
Dim xmlr As XmlTextReader
xmlr = new XmlTextReader(Server.MapPath("nomeFile.xml"))
documento.Load(xmlr)
```

oppure

```
Dim documento As XmlDocument
documento = new XmlDocument()
documento.Load(Server.MapPath("nomeFile.xml"))
```

Poi si possono usare i metodi per navigare o modificare l'albero.

Invece di *XmlDocument* si può usare ***XmlDataDocument*** che ha la proprietà *DataSet* che permette di accedere a una rappresentazione del file XML come *DataSet*.

```
Dim documento As XmlDataDocument
documento = new XmlDataDocument()
documento.DataSet.ReadXml(Server.MapPath("nomeFile.xml"))
```

VERIFICA

QUESTIONARIO

1. Che cos'è un parser XML?
2. Che cos'è l'API SAX e quale procedimento usa?
3. Quali elaborazioni si possono fare su un documento XML usando SAX?
4. Che cos'è l'API DOM e quale procedimento usa?
5. Quali elaborazioni si possono fare su un documento XML usando DOM?
6. Qual è la gerarchia degli oggetti di un documento XML?
7. Come si procede per analizzare un documento XML con DOM?
8. Come si procede per creare un nuovo documento XML con DOM?
9. Come si può salvare un documento XML in un file?
10. Come si può applicare un foglio stile XSL a un documento XML?
11. Come si può generare un documento XML in una pagina JSP?
12. Come si può elaborare un documento XML in una pagina JSP?
13. Come si può applicare un foglio stile XSL a un documento XML in una pagina JSP?
14. Come funziona il controllo Web server *Xml* in ASP.NET?
15. A che cosa servono le classi *XmlTextReader* e *XmlTextWriter*?
16. Come funziona il controllo Web server *XmlDataSource* in ASP.NET?
17. Come si può usare un oggetto *DataSet* per trasformare un database in un file XML e viceversa?
18. A che cosa servono le classi *XmlDocument* e *XmlDataDocument*?

Loescher Editore - Vietata la vendita e la diffusione

Test

1 ..... *Indicare se le seguenti affermazioni sono vere o false.*

V	F	
☐	☐	1) L'API SAX permette di fare modifiche al documento.
☐	☐	2) L'API DOM permette di fare modifiche al documento.
☐	☐	3) L'API SAX permette di creare un nuovo documento.
☐	☐	4) L'API DOM permette di creare un nuovo documento.
☐	☐	5) L'API SAX crea in memoria l'albero degli oggetti del documento.
☐	☐	6) L'API DOM crea in memoria l'albero degli oggetti del documento.
☐	☐	7) Il metodo *writeToString()* di *LSSerializer* scrive un documento XML in un file.
☐	☐	8) Il metodo *transform()* della classe *Transformer* permette di applicare un foglio stile XSL a un documento XML.

Esercizi

1 ..... Dato un articolo memorizzato in un documento XML del tipo

```
<articolo>
    <titolo>...</titolo>
    <autore>...</autore>
    <data>...</data>
    <testo>...</testo>
</articolo>
```

visualizzare l'articolo mediante:

- un'applicazione Java che usi DOM;
- un'applicazione Java che applichi un foglio stile XSL;
- una pagina JSP che usi DOM;
- una pagina JSP che applichi un foglio stile XSL;
- una pagina ASP.NET che usi il controllo *Xml*;
- una pagina ASP.NET che usi la classe *XmlTextReader*;
- una pagina ASP.NET che usi il controllo *XmlDataSource*.

2 ..... Dato un messaggio di posta memorizzato in un documento del tipo:

```
<email>
    <mittente>...</mittente>
    <destinatario>...</destinatario>
    <cc>...</cc>
    ... altri elementi cc
    <bcc>...</bcc>
    ... altri elementi bcc
    <allegato>...</allegato>
    ... altri elementi allegato
    <oggetto>...</oggetto>
    <testo>...</testo>
</email>
```

visualizzare il messaggio mediante:

- un'applicazione Java che usi DOM;

Loescher Editore - Vietata la vendita e la diffusione

- un'applicazione Java che applichi un foglio stile XSL;
- una pagina JSP che usi DOM;
- una pagina JSP che applichi un foglio stile XSL;
- una pagina ASP.NET che usi il controllo *Xml*;
- una pagina ASP.NET che usi la classe *XmlTextReader*;
- una pagina ASP.NET che usi il controllo *XmlDataSource*.

3 ..... Generare tramite una pagina JSP un documento XML del tipo:

```
<?xml version="1.0" ?>
<anagrafica>
    <persona professione="" annoNascita="">
        <nominativo>
            <cognome>...</cognome>
            <nome>...</nome>
        </nominativo>
        <indirizzo>
            <via>...</via>
            <citta>...</citta>
        </indirizzo>
    </persona>
    ... altri elementi persona
</anagrafica>
```

4 ..... Scrivere in un documento XML i dati (predefiniti) di un elemento persona come definito nell'esercizio precedente con:
- un'applicazione Java (si veda l'esempio 12.1);
- una pagina JSP;
- una pagina ASP.NET (si veda l'esempio 12.8).

5 ..... Modificare l'esercizio precedente in modo che i dati di un libro vengano inseriti tramite un modulo e che si possano inserire i dati di tutti i libri desiderati con:
- un'applicazione Java;
- una pagina JSP;
- una pagina ASP.NET.

6 ..... Dati un documento XML e alcuni fogli stile che ne diano visualizzazioni diverse, fare in modo che sia possibile scegliere il foglio stile da applicare con:
- un'applicazione Java;
- una pagina JSP;
- una pagina ASP.NET.

7 ..... Usando una pagina JSP o ASP.NET visualizzare in una tabella i dati di un documento XML; fare in modo che sia possibile ordinare i dati in base ai valori di una colonna facendo clic sull'intestazione della colonna. (Suggerimento: utilizzare un foglio stile per visualizzare i dati ordinati in base al valore di una colonna).

8 ..... Data una tabella di un database trasformarla in un documento XML con:
- un'applicazione Java;
- una pagina JSP;
- una pagina ASP.NET.

9 ..... Dato un documento XML trasformarlo in una tabella di un database con:
- un'applicazione Java;
- una pagina JSP;
- una pagina ASP.NET.

Loescher Editore - Vietata la vendita e la diffusione

PROPOSTE DI LAVORO

Sviluppare le seguenti proposte con applicazioni Java, pagine JSP o pagine ASP.NET, eventualmente realizzando opportuni fogli stile XSL.

1 ..... Dato un documento che descrive una serie di messaggi del tipo:

```
<messaggi>
    <messaggio>
        <autore>...</autore>
        <oggetto>...</oggetto>
        <testo>...</testo>
    </messaggio>
    ... altri elementi messaggio
<messaggi>
```

- visualizzare i messaggi;
- selezionare e visualizzare i messaggi di un certo autore.

2 ..... Dato un documento XML che contenga una serie di vocaboli in italiano con la corrispondente traduzione in inglese:

- visualizzare l'elenco di tutte le parole con la relativa traduzione;
- visualizzare l'elenco delle parole che iniziano con una lettera specificata;
- visualizzare la traduzione di una parola;
- presentare una parola alla volta, in una sola lingua e richiedere all'utente la traduzione, contando le risposte esatte.

3 ..... Gestire le seguenti possibilità:

- scrivere messaggi in un documento XML del tipo:

```
<messaggi>
    <messaggio numero="n">
        testo del messaggio
    </messaggio>
    … altri elementi messaggio
</messaggi>
```

- visualizzare tutti i messaggi;
- visualizzare uno specifico messaggio;
- sostituire il testo di un messaggio;
- cancellare un messaggio.

4 ..... Gestire le seguenti possibilità:

- inserire in un documento XML un messaggio di posta del tipo:

```
<email>
    <mittente>...</mittente>
    <destinatario>...</destinatario>
    <cc>...</cc>
    ... altri elementi cc
    <bcc>...</bcc>
    ... altri elementi bcc
    <allegato>...</allegato>
    ... altri elementi allegato
    <oggetto>...</oggetto>
    <testo>...</testo>
</email>
```

Loescher Editore - Vietata la vendita e la diffusione

- visualizzare il messaggio inserito;
- modificare i dati;
- cancellare gli elementi *cc*, *bcc* e *allegato*.

5 ..... Gestire una semplice rubrica telefonica del tipo:

```
<rubrica>
    <persona>
        <nominativo>Cognome Nome</nominativo>
        <telefono>numero telefonico</numero>
    </persona>
    .. altri elementi persona
</rubrica>
```

deve essere possibile:

- visualizzare tutti i nominativi presenti;
- cercare il numero di una persona;
- inserire un nuovo nominativo (inserendolo al posto giusto in ordine alfabetico);
- modificare il numero di telefono di una persona;
- eliminare un nominativo.

6 ..... Gestire un'agenda del tipo:

```
<agenda anno="aaaa">
    <mese valore="mm">
        <giorno valore="gg">
            <appuntamento dalle="hhmm" alle="hhmm">Descrizione</appuntamento>
            ... altri elementi appuntamento
        </giorno>
        ... altri elementi giorno
    </mese>
    ... altri elementi mese
</agenda>
```

deve essere possibile:

- scegliere una data e un intervallo orario e visualizzare gli impegni relativi;
- inserire un nuovo appuntamento specificando data e ora; l'appuntamento va inserito nel file nell'ordine corretto; se l'ora di inizio precede la fine di un altro appuntamento o l'ora di fine segue l'ora di inizio di un altro appuntamento lo si deve segnalare e permettere l'inserimento solo se espressamente confermato;
- modificare un appuntamento;
- cancellare un appuntamento.

7 ..... Trasformare il documento:

```
<agenda anno="aaaa">
    <mese valore="mm">
        <giorno valore="gg">
            <appuntamento dalle="hhmm" alle="hhmm">Descrizione</appuntamento>
            ... altri elementi appuntamento
        </giorno>
        ... altri elementi giorno
    </mese>
    ... altri elementi mese
</agenda>
```

nel documento:

```
<agenda>
```

Loescher Editore - Vietata la vendita e la diffusione

```
        <anno="aaaa">
            <data mese="mm" giorno="gg">
                <appuntamento>
                    <inizio>...</inizio>
                    <fine>...</fine>
                    <descrizione>...</descrizione>
                </appuntamento>
                ... altri elementi appuntamento
            </data>
            ... altri elementi data
        </anno>
    <agenda>
```

Realizzare i documenti XML, i fogli stile XSL e CSS e le applicazioni Java, le pagine JSP o le pagine ASP.NET necessarie a sviluppare i seguenti progetti.

8 ..... Dato un file XML con le domande e risposte di un test, con la possibilità di scegliere più risposte per ogni domanda, realizzare un'applicazione che permetta di proporre ogni domanda con le possibili risposte e di scegliere le risposte. Memorizzare in un documento XML con una opportuna struttura le risposte scelte da più persone. Utilizzare questo file per effettuare una statistica sulle risposte scelte.

9 ..... Gestione di un forum di messaggi.
(Suggerimento: i documenti XML possono essere del tipo:

forums.xml gruppi disponibili

```
<?xml version = "1.0"?>
<forum>
    <gruppo nomefile = "gruppo1.xml">
        <nome>Nome del primo gruppo</nome>
    </gruppo>
    ... altri elementi gruppo
</forum>
```

gruppo1.xml messaggi di un gruppo

```
<?xml version = "1.0"?>
<gruppo file = "gruppo1.xml">
    <nome> Nome del primo gruppo</nome>
    <messaggio dataora = "aa/mm/gg hh:mm">
        <mittente>Mittente del messaggio</mittente>
        <oggetto>Oggetto del messaggio</oggetto>
        <testo>Testo del messaggio</testo>
    </messaggio>
    ... altri elementi messaggio
</gruppo>
```

10 ..... Gestione di una rubrica telefonica.

11 ..... Gestione di un vocabolario inglese/italiano.

12 ..... Gestione di un glossario.

13 ..... Gestione di un test con domande di tipo vero/falso.

14 ..... Gestione di un test a risposta multipla.

15 ..... Gestione di una biblioteca.

Loescher Editore - Vietata la vendita e la diffusione

Modulo 6

Progetti

13 Temi d'esame

14 Esempi di svolgimento

Loescher Editore - Vietata la vendita e la diffusione

13 TEMI D'ESAME

TRACCIA DI SOLUZIONE

Il punto di partenza di ogni progetto è una situazione reale, più o meno complessa, per cui viene richiesto di realizzare un sistema informatico basato su database.

Bisogna descrivere il sistema in grado di realizzare gli obiettivi richiesti definendo il database e le applicazione necessarie allo scopo.

Uno schema generale di soluzione comprende:

- analisi della realtà di riferimento con eventuali ipotesi aggiuntive e/o semplificative;
- definizione dello schema concettuale del database facendo uso di diagrammi ER o UML ed esplicitando gli eventuali vincoli;
- definizione dello schema relazionale del database evidenziando le chiavi di ogni relazione e i vincoli di integrità referenziale e descrivendo anche il formato dei dati e il significato di ciascun attributo;
- scrittura dei comandi SQL di definizione dei dati;
- scrittura dei comandi di manipolazione e delle query richieste dal problema;
- descrizione dell'applicazione che interagisce con il database; in genere si tratta di applicazioni server side;
- codifica in un linguaggio di programmazione a scelta di un segmento dell'applicazione; in genere vengono richieste (eventualmente in alternativa) le funzioni più significative.

È possibile proporre soluzioni alternative che facciano uso di documenti XML invece che di database.

Qualche volta il punto da sviluppare è preceduto da una richiesta di discussione di un argomento teorico introduttivo; in tal caso è bene concentrarsi sui punti salienti dell'argomento evitando di dilungarsi inutilmente con eccessivi dettagli.

Loescher Editore - Vietata la vendita e la diffusione

1 ..... *Esame di Stato dell'ITIS a indirizzo informatico - Corso di ordinamento - 2008*
Prova scritta di Informatica generale e applicazioni tecnico-scientifiche

Olimpiadi di informatica

In occasioni delle Olimpiadi Internazionali di Informatica 2008, la società organizzatrice desidera realizzare un sistema informatico per la gestione delle gare e degli "atleti".

La base di dati deve consentire la memorizzazione delle informazioni

- degli atleti, che possono partecipare alle gare sia singolarmente sia raggruppati in squadre
- delle gare nelle varie fasi
- delle sedi di gara

Le Olimpiadi prevedono una fase scolastica (in ciascun istituto scolastico partecipante), una fase regionale, una finale nazionale e la gara internazionale che designerà il vincitore e la squadra vincitrice.

Il candidato, fatte le opportune ipotesi aggiuntive, realizzi:

1. un'analisi della realtà di riferimento individuando le possibili soluzioni e scelga quella che a suo motivato giudizio è la più idonea a rispondere alle specifiche indicate
2. uno schema concettuale della base di dati
3. uno schema logico della base di dati
4. la definizione delle relazioni della base di dati in linguaggio SQL
5. le seguenti interrogazioni espresse in linguaggio SQL:
   - stampare l'elenco degli atleti raggruppati per squadre per ogni singola fase
   - dato il nome di un atleta stampare i risultati ottenuti nelle diverse gare alle quali ha partecipato
   - stampare il calendario delle gare
   - stampare una scheda informativa (cognome, nome, istituto scolastico di provenienza, nazionalità) del vincitore e della squadra vincitrice
   - stampare la classifica per ciascuna gara (a parità di punteggio vengono privilegiati gli atleti più giovani)
   - aggiornare, per ciascuna fase (scolastica-regionale-nazionale-internazionale) gli eventuali punteggi record
   - calcolare il punteggio medio ottenuto durante la prima selezione, per ciascun istituto scolastico
   - stampare per ciascuna squadra il numero di "atleti" partecipanti e l'età media
6. l'interfaccia utente che il candidato intende proporre per interagire con la base di dati e codificare in un linguaggio di programmazione a scelta un segmento significativo del progetto realizzato.
7. un sito Internet che presenti al pubblico le classifiche delle diverse gare.

2 ..... *Esame di Stato dell'ITC a indirizzo programmatori - Progetto Mercurio e Corso di ordinamento - 2007*
Prova scritta di Informatica

Trattamento di dati sensibili

Il Codice della Privacy prevede una serie di adempimenti per i soggetti pubblici e

Loescher Editore - Vietata la vendita e la diffusione

privati che trattano dati sensibili con supporti informatici. Il candidato illustri le possibili procedure per:

- garantire l'accesso ai documenti soltanto a soggetti autorizzati;
- gestire i *backup* periodici dei soli documenti contenenti dati sensibili;
- simulare periodicamente la perdita dei dati e il recupero degli stessi dalle copie di *backup*;
- modificare periodicamente i parametri di accesso ai documenti protetti.

Successivamente sviluppi il seguente punto.

Uno studio medico associato, in cui operano medici di base e medici specialisti, dà incarico a una società d'informatica di progettare e realizzare un database contenente i dati anagrafici e professionali dei medici, l'orario delle visite, i tempi medi previsti per ogni visita, il costo delle singole prestazioni, i dati anagrafici dei pazienti e le prestazioni richieste.

Elaborare, dopo aver descritto le caratteristiche dell'applicativo che si intende utilizzare, il progetto definendone i dati necessari e dettagliando la procedura per risolvere la prima e due richieste tra quelle indicate:

1. elenco giornaliero delle visite prenotate per ogni singolo medico;
2. elenco giornaliero delle visite prenotate e non effettuate;
3. elenco settimanale contenente gli appuntamenti di ciascun medico suddivisi per giorno e per ora;
4. elenco cronologico delle visite usufruite da ciascun paziente.

Realizzare, infine, la pagina *web* con la quale lo studio medico pubblicizza la propria attività fornendo l'indicazione dei servizi e il quadro orario.

Dati mancanti opportunamente scelti.

3 ..... *Esame di Stato dell'ITIS a indirizzo informatico - Progetto Abacus e Corso di ordinamento - 2007*

Prova scritta di Informatica

Portale per la vendita di libri

Un negozio on line chiede che sia progettato e realizzato un database per l'organizzazione e la gestione di un portale per la vendita di libri su Internet.

Il negozio richiede che:

- il Catalogo dei libri sia organizzato per:
  - Reparti (in ordine alfabetico: *architettura e urbanistica, arte, classici greci e latini, cucina e casa, diritto, economia e management, fantascienza e fantasy, filosofia, fumetti, ...*)
- ciascun Reparto sia organizzato in libri:
  - "Da non perdere" caratterizzati da uno sconto del 20% sul prezzo di copertina
  - "I più venduti" caratterizzati dal maggior numero di copie vendute negli ultimi 30 giorni
  - "Ultimi arrivi" caratterizzati dai titoli aggiunti negli ultimi 30 giorni
  - "Offerte speciali" caratterizzate da uno sconto del 25% sul prezzo di copertina
  - "Remainders" caratterizzati da uno sconto del 50% sul prezzo di copertina

Loescher Editore - Vietata la vendita e la diffusione

- ciascun libro in negozio sia archiviato con le seguenti ulteriori informazioni:
  - titolo, autore, prezzo di copertina, sconto (eventuale), dati (numero di pagine, rilegato, ...), anno di pubblicazione, editore, collana, immagine (foto della copertina), data di archiviazione
- gli utenti abilitati all'acquisto on line siano registrati con i seguenti dati:
  - nome e cognome, indirizzo, codice di avviamento postale, città, nazione, telefono, fax, e-mail, password, numero di carta di credito, tipo di carta di credito (Visa, CartaSi, Mastercard, ...), data di scadenza della carta di credito
- gli utenti possano chiedere, in fase di acquisto, che il negozio emetta fattura
- il Carrello Acquisti sia organizzato come segue:
  - numero di copie acquistate, titolo, autore, prezzo, disponibilità (giorni, settimane, ...), togli dal carrello (casella di spunta)
- il riepilogo degli ordini sia organizzato come segue:
  - data dell'ordine, numero d'ordine
  - dettagli dell'ordine (riferimento articolo, quantità, titolo, prezzo di copertina, prezzo scontato, stato dell'ordine, evasione prevista dalla data dell'ordine).

Il candidato, dopo aver fatto le eventuali ipotesi aggiuntive:

a. Fornisca:
   1. lo schema concettuale e lo schema logico del database
   2. la definizione delle relazioni in linguaggio SQL.

b. Implementi le seguenti query:
   1. *Ricerca per Reparto*: scelto un Reparto, il numero di libri "Da non perdere" ed il numero di libri "I più venduti" con i relativi dettagli
   2. *Ordini*: gli ordini in corso, con dettagli, di un dato utente.

c. Scriva in un linguaggio lato server, il codice di almeno una delle seguenti pagine del portale:
   1. con accesso libero, la pagina utile a visualizzare i Reparti e, per ciascun reparto, la pagina che implementa la query n. 1;
   2. con accesso riservato agli utenti registrati, la composizione degli ordini in corso.

4 ..... *Esame di Stato dell'ITIS a indirizzo informatico - Corso di ordinamento - 2006*
Prova scritta di Informatica generale e applicazioni tecnico-scientifiche

Community di apprendimento sul Web

Una rete di scuole chiede che sia progettato e realizzato un database per l'organizzazione e la gestione del portale di una community di apprendimento sul Web.

L'organizzazione della community prevede che:

- l'accesso sia consentito ai soli utenti registrati;
- gli utenti siano distribuiti in tre gruppi: amministratore, docente, studente tali che:
  - un amministratore abbia accesso a tutte le aree protette del portale;
  - un docente abbia accesso a tutte le aree protette tranne che all'area di amministrazione;
  - uno studente abbia accesso alla propria area protetta e non abbia accesso né all'area di amministrazione né all'area riservata al gruppo docente;

Loescher Editore - Vietata la vendita e la diffusione

- la registrazione degli utenti consenta:
  - alla rete di scuole di acquisire informazioni, sotto forma di dati non sensibili, relative agli utenti quali, ad esempio, nome e cognome, scuola o istituzione formativa di appartenenza, collocazione geografica, e-mail, ecc….
  - agli utenti di scegliere un nome utente, una password e il gruppo di appartenenza tranne quello degli amministratori;
- agli utenti registrati, ciascuno per il proprio gruppo di appartenenza, sia consentito di effettuare l'upload di documenti multimediali archiviando:
  - il titolo
  - il tipo di documento (testo, audio, ecc.)
  - una descrizione sintetica
  - la data di upload
  - i dati personali che lo riguardano;
- il gruppo degli studenti possa usufruire di moduli formativi ad esso rivolti;
- il gruppo dei docenti possa usufruire sia di propri moduli formativi che di quelli rivolti agli studenti;
- ciascun modulo formativo sia individuato da un titolo, da una breve descrizione e dal tipo di utente cui è rivolto.

Il candidato, fatte eventuali ipotesi aggiuntive:

- fornisca:
  1. Lo schema concettuale e lo schema logico del database;
  2. La definizione delle relazioni in linguaggio SQL.
- implementi almeno una delle seguenti query:

  n. 1: I docenti che hanno un account presso la community con la rispettiva collocazione geografica ed i moduli formativi scelti.

  n. 2: I dati relativi agli studenti e ai documenti che essi hanno inviato in remoto sul portale della community mediante upload.
- scriva, in un linguaggio lato server, il codice di almeno una delle seguenti pagine del portale:
  - con accesso riservato all'amministratore, il report che trae i dati dalla query n. 1
  - con accesso riservato agli studenti, il report che trae i dati dalla query n. 2
  - registrazione di un nuovo utente con eventuale invio automatico dei dati registrati mediante e-mail diretta all'utente appena registrato e ad un amministratore.

5 ..... *Esame di Stato dell'ITC a indirizzo programmatori - Progetto Mercurio e Corso di ordinamento - 2005*
Prova scritta di Informatica

Gestione palestra

Telelavoro ed e-commerce, due mini-rivoluzioni, stentano a svilupparsi nel nostro Paese a differenza di quanto è avvenuto negli Stati Uniti e in altre parti d'Europa. Il candidato, dopo aver brevemente illustrato le caratteristiche tecniche di tali processi, analizzi le ragioni del loro limitato utilizzo in Italia e sviluppi il seguente punto.

Una palestra vuole informatizzare la gestione dei corsi offerti alla propria clientela che si caratterizzano per:

Loescher Editore - Vietata la vendita e la diffusione

- la presenza di istruttori specializzati in ciascun corso;
- il numero massimo di partecipanti variabile per ciascun corso;
- l'offerta dello stesso corso in diversi orari della giornata;
- la partecipazione al corso con idoneo certificato medico di durata annuale.

Organizzare un database in grado di fornire le seguenti informazioni:

1. elenco degli iscritti a ogni corso
2. numero dei posti liberi in ciascun corso
3. elenco dei corsi tenuti da ogni istruttore
4. elenco dei clienti con certificato medico scaduto

e, dopo aver

- descritto le caratteristiche dell'applicativo che si intende utilizzare,
- elaborato il progetto dell'applicazione complessiva, comprendente l'analisi, lo schema, la definizione dei dati e il menu di scelta,

realizzare almeno uno dei quattro punti sopra indicati utilizzando l'applicativo descritto.

Progettare, infine, la pagina web con la quale la palestra presenta alla clientela i corsi, le relative caratteristiche e gli orari.

Dati mancanti opportunamente scelti.

6 ..... *Esame di Stato dell'ITIS a indirizzo informatico - Progetto Abacus e Corso di ordinamento - 2005*
Prova scritta di Informatica
Casa discografica
Una etichetta discografica indipendente chiede che sia realizzata una base di dati utile all'archiviazione ed alla gestione delle proprie produzioni musicali, anche al fine di renderle eventualmente disponibili su Internet.
Le scelte editoriali dell'etichetta sono relative a diversi generi musicali ciascuno comprendente diversi sottogeneri come, ad esempio:

1. Genere Popolare	2. Genere classico
Sottogenere	Sottogenere
a) Avanguardia	a) Musica da camera
b) Blues	b) Concerto
c) Elettronica	c) Opera
d) Folk	d) Sinfonia
e) Jazz	e) Musica corale
f) New Age	f) ……..
g) Rap	
h) Rock	
i) ……	

L'organizzazione della casa discografica prevede che:

- un artista o un gruppo musicale o un'orchestra possano registrare musica di generi diversi o, nell'ambito dello stesso genere, di diversi sottogeneri;
- possano essere rilasciate più edizioni di uno stesso album ciascuna con una diversa copertina;

Loescher Editore - Vietata la vendita e la diffusione

- un artista o un gruppo o un'orchestra possano non essere più attivi sul mercato discografico.

Si chiede, in particolare, che si possano archiviare i seguenti dati:

a) il nome dell'artista e/o del gruppo musicale e/o dell'orchestra;
b) il periodo (in anni) di attività sul mercato discografico;
c) il titolo dell'album;
d) il nome del direttore d'orchestra;
e) la data (giorno - mese – anno) di pubblicazione della/e edizione/i;
f) il titolo dei brani musicali presenti nell'album;
g) la durata (minuti – secondi) dell'album;
h) la durata (minuti – secondi) dei singoli brani dell'album;
i) il nome dell'autore o degli autori dei brani;
j) brevi notizie biografiche dell'artista e/o del gruppo musicale e/o del direttore d'orchestra;
k) l'immagine della copertina dell'album.

Il candidato, fatte le opportune ipotesi aggiuntive, progetti una base di dati utile alla realizzazione dell'archivio richiesto dall'etichetta indipendente, fornendo:

1. uno schema concettuale della base di dati;
2. uno schema logico della base di dati;
3. la definizione delle relazioni della base di dati in linguaggio SQL;

ed inoltre:

4. implementi in linguaggio SQL la seguente interrogazione:
   - Dato il titolo di un brano, quali sono le date di pubblicazione, la durata, l'artista e/o il gruppo musicale e/o l'orchestra con il relativo periodo di attività nonché gli autori e l'album di appartenenza.

Opzionalmente, supponendo che la base di dati sia disponibile su di un server on line, il candidato implementi una directory per il web che consenta, a partire da una pagina con l'elenco dei generi musicali, di ottenere:

- con un click sul nome di un genere musicale, una pagina con l'elenco dei nomi dei sottogeneri musicali;
- con un click sul nome di un sottogenere musicale, una pagina con l'elenco dei nomi degli artisti;
- con un click sul nome di un artista, l'elenco degli album di quell'artista con le immagini di copertina, la durata, l'anno o gli anni di edizione, le notizie bibliografiche sull'artista ed il periodo di attività.

7 ..... *Esame di Stato dell'ITIS a indirizzo informatico - Progetto Abacus e Corso di ordinamento - 2004*
Prova scritta di Informatica
Attività scolastiche ed extrascolastiche
Il Dirigente Scolastico di una Scuola Secondaria Superiore chiede che si realizzi una base di dati per l'archiviazione e la gestione di informazioni riguardanti le attività scolastiche ed extrascolastiche documentate, nonché i risultati scolastici conseguiti da ciascuno studente al fine di produrre, in itinere e/o al termine del corso di studi, un portfolio studente.

Loescher Editore - Vietata la vendita e la diffusione

L'organizzazione scolastica dell'istituto prevede che:

a) ciascuno studente possa frequentare più corsi di recupero e/o di sostegno e/o di arricchimento dell'offerta formativa;

b) ogni corso abbia un titolo, una descrizione, una data di inizio e di fine, un monte ore definito;

c) ogni studente possa frequentare più corsi esterni alla scuola;

d) i corsi esterni alla scuola hanno un titolo, una descrizione, una data di inizio e di fine, un monte ore definito, un riferimento che indichi l'Ente e/o l'Istituzione che li ha organizzati, un riferimento alla documentazione di accertamento;

e) ogni classe sia individuata univocamente da un numero ordinale progressivo (I, II, III, IV, V) e da una lettera che ne indica la sezione di appartenenza (A, B, C, D, E, F, G,).

In particolare, il Dirigente Scolastico chiede che si possa procedere all'archiviazione dei:

- dati anagrafici degli studenti utili alla loro univoca identificazione;
- dati relativi alla frequenza delle classi del corso di studi (quali classi ciascuno studente ha frequentato in ordine crescente ed in quale anno scolastico);
- dati relativi agli esiti conclusivi di ciascun anno scolastico (promozione sì/no, eventuali debiti formativi ed in quale disciplina) per ciascuno studente;
- dati relativi ai corsi interni;
- dati relativi ai corsi esterni alla scuola purché documentati.

Il candidato, fatte le opportune ipotesi aggiuntive, progetti una base di dati utile alla realizzazione del portfolio studente richiesto dal Dirigente Scolastico, fornendo:

1. uno schema concettuale della base di dati;
2. uno schema logico della base di dati;
3. la definizione delle relazioni della base di dati in linguaggio SQL;

ed inoltre:

4. implementi in linguaggio SQL le seguenti interrogazioni:
   - data una classe, quali studenti di quella classe hanno frequentato corsi e di che tipo;
   - dato uno studente, quali corsi ha frequentato, di che tipo, per quale monte ore e in quale anno scolastico;
   - dato un anno scolastico, quali corsi interni sono stati attivati e da quali studenti sono stati seguiti;
   - dato un corso, quali sono i dati relativi ad esso e per quali anni scolastici è stato attivato;
   - dato uno studente quali classi ha frequentato, in quali anni scolastici e con quali esiti finali.

8 ..... *Esame di Stato dell'ITC a indirizzo programmatori - Progetto Mercurio - 2003*
Prova scritta di Informatica

Corsi di inglese online

Alcuni esperti prevedono che il moltiplicarsi degli utenti e delle transazioni metterà a dura prova i gestori di rete e le infrastrutture legate a Internet. Il candidato

Loescher Editore - Vietata la vendita e la diffusione

presenti le proprie riflessioni sulle implicazioni determinate da tale fenomeno e descriva le apparecchiature necessarie per assicurare la piena funzionalità dei servizi on line.
Successivamente sviluppi il seguente caso.

Un Istituto specializzato nell'insegnamento dell'inglese organizza corsi on-line su Internet adottando metodologie di apprendimento a distanza. I corsi sono strutturati su più livelli (1: principiante, 2: intermedio, 3: esperto) e dopo l'iscrizione l'allievo deve sostenere un test d'ingresso con quesiti a risposta chiusa che misurano il grado di conoscenza della lingua inglese. L'inserimento in uno o nell'altro dei tre livelli dipenderà dal numero di risposte esatte sul totale di quelle proposte nel test. I risultati vengono registrati in un'apposita base di dati.
Alla fine del corso l'allievo deve sostenere un test finale.

Il candidato, dopo aver:

- indicato la caratteristiche dell'applicativo che intende utilizzare,
- ipotizzato e rappresentato lo schema generale della procedura,
- indicato come organizzare i dati,

sviluppi uno dei seguenti punti in un linguaggio a sua scelta e con dati opportunamente assunti.

1. Perfezionare l'iscrizione di un allievo sulla base del risultato raggiunto al test (inserimento nel livello corrispondente).
2. Stampare l'elenco degli allievi di un corso che hanno superato il test finale.
3. Realizzare le pagine Web idonee a gestire la somministrazione del test d'ingresso e l'eventuale interazione con la base di dati.

9 ..... *Esame di Stato dell'ITC a indirizzo programmatori - Corso di ordinamento - 2003*
Prova scritta di Informatica gestionale

Istituto di credito

Le potenzialità offerte dalla diffusione della telematica sono state colte anche dalle aziende di credito al fine di rendere più efficace e più economica la gestione delle attività bancarie. Il candidato, dopo aver illustrato e commentato le più significative procedure telematiche del settore, svolga il seguente caso.

Un istituto di credito intende istituire un servizio on line attraverso cui i clienti possano acquisire informazioni, operare su conti correnti, effettuare pagamenti e gestire titoli. Il servizio deve prevedere, fra le altre, le seguenti funzioni:

a. creazione e gestione di un archivio che fornisca informazioni sull'andamento dei titoli,
b. visualizzazione e stampa dell'estratto conto,
c. gestione dei mandati di pagamento.

Il candidato, dopo aver:

- indicato le caratteristiche dell'applicativo che intende utilizzare,
- ipotizzato e rappresentato lo schema generale della procedura,
- indicato come intende organizzare i dati,

sviluppi uno dei seguenti punti in un linguaggio a sua scelta e con dati opportunamente assunti.

Loescher Editore - Vietata la vendita e la diffusione

1. Produrre la stampa dell'estratto conto.
2. Visualizzare l'andamento di un titolo in un certo periodo.
3. Realizzare le pagine Web idonee a presentare i servizi offerti dalla banca.

10 ..... *Esame di Stato dell'ITIS a indirizzo informatico - Progetto Abacus e Corso di ordinamento - 2003*
Prova scritta di Informatica
Gestione vivaio
Un vivaio vuole realizzare una base di dati per gestire le sue attività di vendita di piante e le sue attività esterne.
Si vogliono memorizzare, oltre alle informazioni generali sulle diverse specie di piante, anche quelle relative alle specifiche piante presenti nel vivaio.
Di ogni specie deve essere registrato il nome, un'immagine, una breve descrizione, informazioni relative al modo di coltivazione e alle caratteristiche dell'esposizione ed infine se si tratta di pianta da interno o da esterno.
Si vogliono inoltre registrare i dati relativi alle piante effettivamente presenti nel vivaio, raggruppate per specie, eventualmente suddivise in diversi tipi in base al prezzo di vendita. Per ogni specie (o per ogni tipo, se la specie è suddivisa in tipi), è necessario registrare il numero di esemplari presenti nel vivaio e il costo di ogni esemplare.
Si vogliono inoltre gestire le informazioni relative al personale che lavora nel vivaio (agronomi, operai, amministrativi) registrando i dati anagrafici, la qualifica e, per gli agronomi, l'anno di assunzione nel vivaio. Ogni singola specie del vivaio è sotto la responsabilità di un agronomo.
Le attività esterne del vivaio, quali ad esempio la potatura o la manutenzione dei giardini, sono svolte dal personale in base alla qualifica (in generale ogni lavoratore è in grado di svolgere più di un'attività) e sono caratterizzate da un codice, da un nome, da un costo orario.

Per le attività esterne si vogliono mantenere tutte le informazioni relative ai clienti che richiedono le attività, in particolare se si tratta di privati o di aziende, la data di prenotazione e quella di effettuazione dell'intervento richiesto e se per l'intervento sono necessarie piante del vivaio.

Il candidato consideri la situazione sopra descritta, precisi eventuali ipotesi aggiuntive e realizzi:

- un'analisi della realtà di riferimento che illustri le premesse per i successivi passi della progettazione della base di dati;
- uno schema concettuale della base di dati;
- uno schema logico della base di dati;
- la definizione delle relazioni della base di dati in linguaggio SQL;
- le seguenti interrogazioni espresse in linguaggio SQL:
  - dato il nome di una pianta, riportare quanti esemplari di quella pianta sono presenti nel vivaio;
  - dato il nome di una stagione, visualizzare il nome delle piante che fioriscono in quella stagione;
  - dato il nome di un intervento esterno, tra quelli previsti dal vivaio, riportare il nome e il telefono dei soggetti che hanno richiesto quell'intervento nel corso di un determinato anno solare;

Loescher Editore - Vietata la vendita e la diffusione

- dato il nome di un agronomo, riportare quanti esemplari di piante sono sotto la sua responsabilità;
- visualizzare nome, descrizione e quantità di esemplari presenti nel vivaio, della pianta più economica da interno;
- riportare nome degli interventi richiesti non ancora evasi con il nome e il telefono del richiedente.

Il candidato sviluppi inoltre, a scelta uno dei seguenti moduli:

- Si vuole realizzare un sito Internet che presenti al pubblico il vivaio illustrandone i prodotti e le diverse attività.
- Si vuole consentire la gestione delle attività esterne del vivaio attraverso una prenotazione on-line da parte dei clienti. Illustrare le modalità di realizzazione di questa funzione e gli strumenti tecnici adottabili.

11 ..... *Esame di Stato dell'ITIS a indirizzo informatico - Progetto Abacus - 2002*
Prova scritta di Informatica

Banca del Tempo

Un'associazione "Banca del Tempo" vuole realizzare una base di dati per registrare e gestire le attività dell'associazione.
La "Banca del Tempo" (BdT) indica uno di quei sistemi organizzati di persone che si associano per scambiare servizi e/o saperi, attuando un aiuto reciproco.
Attraverso la BdT le persone mettono a disposizione il proprio tempo per determinate prestazioni (effettuare una piccola riparazione in casa, preparare una torta, conversare in lingua straniera, ecc.) aspettando di ricevere prestazioni da altri.
Non circola denaro, tutte le prestazioni sono valutate in tempo, anche le attività di segreteria.
Le prestazioni sono suddivise in categorie (lavori manuali, tecnologie, servizi di trasporto, bambini, attività sportive, ecc.).
Chi dà un'ora del suo tempo a qualunque socio, riceve un'ora di tempo da chiunque faccia parte della BdT. La base di dati dovrà mantenere le informazioni relative ad ogni prestazione (quale prestazione, da chi è stata erogata, quale socio ha ricevuto quella prestazione, per quante ore e in quale data) per consentire anche interrogazioni di tipo statistico.
Il territorio di riferimento della BdT è limitato (un quartiere in una grande città o un piccolo comune) ed è suddiviso in zone; la base di dati dovrà contenere la mappa del territorio e delle singole zone, in forma grafica.

Si consideri la realtà di riferimento sopra descritta e si realizzino:

1. la progettazione concettuale della realtà indicata attraverso la produzione di uno schema (ad esempio ER, Entity-Relationship) con gli attributi di ogni entità, il tipo di ogni relazione e i suoi eventuali attributi;
2. una traduzione dello schema concettuale realizzato in uno schema logico (ad esempio secondo uno schema relazionale;
3. le seguenti interrogazioni espresse in algebra relazionale e/o in linguaggio SQL:
   a. produrre l'elenco dei soci (con cognome, nome e telefono) che hanno un "debito" nella BdT (coloro che hanno usufruito di ore di prestazioni in numero superiore a quelle erogate);
   b. data una richiesta di prestazione, visualizzare la porzione di mappa del territorio nel quale si trova il socio richiedente e l'elenco di tutti i soci che si

Loescher Editore - Vietata la vendita e la diffusione

trovano in quella zona in grado di erogare quella prestazione, visualizzandone il nome, cognome, indirizzo e numero di telefono;

c. visualizzare tutti i soci che fanno parte della segreteria e che offrono anche altri tipi di prestazione;

d. produrre un elenco delle prestazioni ordinato in modo decrescente secondo il numero di ore erogate per ciascuna prestazione.

4. (Facoltativo) Sviluppare il problema posto scegliendo una delle due seguenti proposte descrivendone le problematiche e le soluzioni tecniche adottabili:

4.1 l'associazione BdT vuole realizzare un sito Web per rendere pubbliche le sue attività consentendo anche di effettuare on-line le interrogazioni della base di dati previste nel punto 3;

4.2 l'associazione BdT vuole realizzare un sito Web attraverso il quale possa raccogliere l'adesione on-line di altri associati, attraverso il riempimento di un modulo da inviare via Internet all'associazione.

12 ..... *Esame di Stato dell'ITC a indirizzo programmatori - Progetto Mercurio - 2001*
Prova scritta di Informatica

Band musicale

Nell'ultimo decennio i cambiamenti prodotti dal diffondersi delle tecnologie informatiche hanno inciso su diversi aspetti della vita sociale. Si sono così affermate non solo nuove figure professionali, ma anche nuove modalità operative che hanno modificato e stanno modificando la vita quotidiana.

Il candidato esponga le proprie considerazioni al riguardo e sviluppi poi il seguente caso.

Una nota band musicale decide di tenere due concerti nella stessa città, a distanza di tre giorni l'uno dall'altro. Il sistema di prenotazioni e acquisto dei biglietti è affidato ad una società che lo gestisce in modo totalmente informatizzato. La gestione del servizio deve prevedere:

- la registrazione della prenotazione a fronte della disponibilità del posto nella data prescelta;
- la registrazione del pagamento con l'emissione del biglietto;
- la segnalazione dell'esaurimento dei posti in una o entrambe le date previste;
- la prenotazione automatica, previo consenso, per l'altra data qualora per quella prescelta risultassero esauriti i posti disponibili;
- la creazione di una pagina Web relativa all'evento in cui compaiano una breve descrizione della band, le date dei concerti, la disponibilità di posti e due immagini fotografiche.

Il candidato dopo aver:

- illustrato le caratteristiche dell'applicativo che intende utilizzare per realizzare il software,
- ipotizzato e rappresentato lo schema generale della procedura,
- indicato come intende organizzare i dati,
- disegnato la pagina Web,
- descritto la sequenza delle operazioni da compiere per la pubblicazione della pagina Web presso un sito ospitante,

Loescher Editore - Vietata la vendita e la diffusione

sviluppi uno dei seguenti punti in un linguaggio a sua scelta:

1. Produrre l'elenco aggiornato delle prenotazioni per ciascuna data.
2. Stampare il biglietto recante l'indicazione della data ed un codice identificativo.
3. Costruire la pagina Web secondo le indicazioni precedentemente esposte.

13 ..... *Esame di Stato dell'ITC a indirizzo programmatori - Corso di ordinamento - 2001*
Prova scritta di Informatica gestionale

Acquisto telematico

La storia di Internet mette in evidenza che il diffondersi delle reti telematiche ha influenzato non solo il campo tecnico della comunicazione tra computer, ma anche le abitudini di vita dei singoli e delle organizzazioni: infatti, l'utilizzo di strumenti *on line* sta diventando di uso comune non solo per l'acquisizione di informazioni, ma anche nello scambio economico e nei rapporti interpersonali.

Il candidato esprima le proprie considerazioni su tali cambiamenti e sviluppi poi il seguente caso.

Una grande società commerciale specializzata nella vendita di apparati elettronici di largo consumo decide di introdurre l'acquisto telematico dei suoi prodotti. La società possiede 20 punti vendita e 3 magazzini di stoccaggio sul territorio nazionale, situati a Milano, Roma e Palermo. Tutte le sedi sono fornite di computer interconnessi in rete dedicata il cui server principale è situato a Milano.
Si vuole che il prodotto ordinato, se disponibile in uno dei punti vendita, sia consegnato presso l'abitazione dell'acquirente in un tempo non superiore alle 24 ore, mentre, se il prodotto è giacente in uno dei 3 magazzini, il tempo massimo di consegna non deve superare le 48 ore.
La scelta e l'ordine di un prodotto dovranno essere gestiti tramite un programma che interagisce con il server di Milano (dove risiede la tabella aggiornata dei prodotti) e fornisca anche le specifiche del prodotto, arricchite da immagini illustrative, la disponibilità, il prezzo, il tempo di consegna.

Il candidato:

- ipotizzi e rappresenti lo schema generale della procedura;
- descriva le modalità e i tempi con cui il database del server viene aggiornato dalle varie sedi e dai tre magazzini;
- indichi come intende organizzare i dati;
- descriva le modalità di implementazione del programma sul sito *Web* della società;

Successivamente sviluppi, in un linguaggio a sua scelta, una delle seguenti parti di programma che:

1. visualizzi i prodotti di una stessa categoria in un intervallo di prezzo prescelto;
2. aggiorni la giacenza dei prodotti in funzione degli ordini evasi;
3. visualizzi l'elenco degli ordini evasi entro le 24 ore dai vari punti vendita con riferimento a un prescelto arco di tempo.

14 ..... *Esame di Stato dell'ITC a indirizzo programmatori - Progetto Mercurio - 2000*
Prova scritta di Informatica

Le tecnologie informatiche e telematiche, in particolare quelle legate a Internet, stanno trasformando rapidamente lo scenario in cui si svolgono le attività aziendali.

Loescher Editore - Vietata la vendita e la diffusione

Dopo aver illustrato come è caratterizzata l'organizzazione di una rete, il candidato descriva schematicamente quali servizi telematici sono necessari alla realizzazione degli scambi via rete (*e-commerce, e-service, ecc.*).

Successivamente sviluppi uno dei seguenti temi.

Tema n. 1
Gestione di un concorso
La società DATA BYTE ha ottenuto una commessa per automatizzare la gestione di un concorso.
Il progetto *software* che si intende realizzare deve essere in grado di acquisire:

- i dati anagrafici dei partecipanti;
- i titoli posseduti (laurea con punti da 1 a 5, pubblicazioni da 0 a 3, specializzazioni da 0 a 3);
- i risultati delle prove (test con punti da 0 a 10, colloquio da 0 a 20);

e di produrre la graduatoria finale. L'elenco di coloro che hanno superato il concorso dovrà essere consultabile in ordine di punteggio e in ordine alfabetico.

Presentare:

- lo schema generale del progetto ipotizzando che i dati possano essere acquisiti anche via *e-mail* e pubblicati su un sito Internet;
- l'organizzazione e la descrizione dei dati;
- uno dei seguenti moduli sviluppato con strumenti informatici a scelta (ambienti applicativi, codifica di programmi ecc.)
  1. elenco dei partecipanti ordinato per titoli;
  2. elenco degli esclusi (motivi esclusione: età>40, nessuna laurea, punteggio prove<20);
  3. stampa della graduatoria finale.

Tema n. 2
Gestione degli ordini
Per attuare il rinnovamento della propria divisione commerciale, un'azienda intende vendere anche direttamente i propri prodotti. Ciò comporta che gli ordini dei clienti possano giungere sia tramite Internet sia tramite telefono presso un centro di raccolta (*call center*).
Tutti gli ordini confluiscono in un sistema che provvede alla loro registrazione e gestione.

Presentare:

- lo schema generale del sistema per la gestione degli ordini;
- l'organizzazione e la descrizione dei dati;
- uno dei seguenti moduli sviluppato con strumenti informatici a scelta (ambienti applicativi, codifica di programmi ecc.)
  1. emissione della fattura relativa all'ordine di un cliente;
  2. elenco dei clienti che hanno ordinato un prodotto specificato in input;
  3. prospetto degli articoli ordinati attraverso Internet in una data specificata.

Loescher Editore - Vietata la vendita e la diffusione

15 ..... *Esame di Stato dell'ITIS a indirizzo informatico - Progetto Abacus e Corso di ordinamento - 2000*
Prova scritta di Informatica
Servizi turistici "Vacanze"
La società di servizi turistici "Vacanze" vuole condurre una indagine sul gradimento dei 10 'pacchetti' di vacanze organizzate che costituiscono il suo catalogo.

Ogni indagine si basa su un questionario a struttura fissa, che viene fatto compilare a un campione di agenzie di viaggi:

- sezione 1: dati anagrafici dell'agenzia (provincia di residenza, n° di dipendenti, n° di offerte di "Vacanze" vendute nell'anno);
- sezione 2: M domande generali su tutto il catalogo a cui l'agenzia intervistata deve rispondere con un valore numerico da 1 a 5;
- sezione 3: N domande specifiche, ripetute per ciascun pacchetto, a cui l'agenzia intervistata deve rispondere con un si o con un no.

Si desidera organizzare l'indagine con l'aiuto di un sistema informatico che supporti le seguenti funzioni:

a) emissione di una lettera rivolta alle agenzie con preghiera di compilazione;
b) compilazione del questionario da parte delle agenzie direttamente mediante la tastiera della stazione di lavoro;
c) raccolta delle risposte e creazione del relativo archivio;
d) analisi dei dati e stampe.

Si dispone dell'archivio delle agenzie da intervistare con i relativi indirizzi postali e di posta elettronica.

Il candidato, fatte tutte le ipotesi aggiuntive che ritiene necessarie

1. indichi come propone di organizzare le funzioni a) b) c) d) e con quali strumenti informatici e programmi intende supportarle;
2. proponga e illustri la struttura degli archivi e lo schema generale del sistema;
3. illustri in particolare la realizzazione di una parte del sistema relativa alle funzioni b) e c).

16 ..... *Esame di Stato dell'ITIS a indirizzo informatico - Progetto Abacus - 1999*
Prova scritta di Informatica
Agenzia immobiliare
Un'agenzia immobiliare vuole realizzare una applicazione software per la realizzazione di un archivio delle case disponibili. Per ogni casa sono disponibili le seguenti informazioni:

- la zona;
- il costo;
- superficie in metri quadri;
- l'anno di costruzione;
- una breve descrizione di non più di cento parole;
- eventualmente una fotografia che non necessariamente esiste per tutte le case.

L'applicazione deve consentire all'operatore almeno le seguenti operazioni:

Loescher Editore - Vietata la vendita e la diffusione

a. introduzione delle informazioni relative a una nuova casa o cancellazione di quelle di una casa non più disponibile;

b. ricerca di una casa in archivio in base a uno p più dei seguenti elementi di ricerca:
   - fascia di prezzo,
   - superficie,
   - zona;

c. visione di tutte le informazioni associate alla casa trovata e, su richiesta, anche della fotografia o una scritta "non disponibile".

Nel caso che la ricerca offra più case con le caratteristiche desiderate deve essere possibile esaminarle una per una.

Il candidato, formulate le ipotesi aggiuntive che ritiene necessarie, deve:

1. illustrare uno schema generale delle procedure per la realizzazione delle diverse funzioni;
2. indicare quali strumenti software intende utilizzare per realizzarle;
3. illustrare come propone di organizzare e rappresentare i dati;
4. realizzare, con lo strumento software prescelto, almeno una parte di una delle due funzioni seguenti:
   - introduzione in catalogo delle informazioni relative a una nuova casa;
   - ricerca in archivio di una casa con le caratteristiche desiderate.

17 ..... *Esame di Stato dell'ITIS a indirizzo informatico - Progetto Abacus - 1998*
Prova scritta di Informatica

Galleria d'arte

Una galleria d'arte ha deciso di creare un sistema che consenta ai suoi clienti di consultare da casa il catalogo: dei quadri, tramite accesso a una pagina web che la galleria può creare presso un fornitore di servizi telematici.

Per ogni quadro è compilata una scheda che riporta l'autore, il titolo, la tecnica (olio, tempera ecc.), le dimensioni, il prezzo. La consultazione del catalogo: può avvenire o semplicemente scorrendo avanti e indietro le schede in ordine alfabetico oppure cercando uno specifico autore.
Il candidato, fatte le ipotesi aggiuntive che ritiene necessarie,

1) proponga una soluzione per la creazione del sistema illustrandone la struttura a blocchi e indicando una soluzione di principio per ciascun blocco;
2) proponga e illustri una struttura per il Data Base dei quadri,
3) sviluppi in dettaglio la soluzione per almeno una delle seguenti funzioni, codificandone un segmento con uno strumento software di sua conoscenza:
   a) creazione del Data Base,
   b) creazione di una semplice pagina web della galleria,
   c) interfaccia per consentire al cliente la consultazione del catalogo e la visione delle singole schede,
4) facoltativamente proponga una soluzione di principio per realizzare un sistema che consenta di mostrare al cliente non solo la scheda di catalogo, ma anche una fotografia del quadro.

Loescher Editore - Vietata la vendita e la diffusione

18 ..... *Esame di Stato dell'ITC a indirizzo programmatori - Corso di ordinamento - 1996*
Prova scritta di Informatica gestionale

Marketing

Il responsabile marketing della società TELELAB, che opera nel settore dei servizi informatici, intende allargare la clientela offrendo la propria collaborazione a imprese italiane del settore merceologico "Y", tale azione potrebbe essere agevolata acquistando un elenco anagrafico, disponibile su supporto informatico, contenente i dati delle imprese di quel settore.

Al fine di illustrare alla Direzione Generale della TELELAB la fattibilità dell'operazione occorre predisporre una relazione che, tra l'altro, precisi:

- quali informazioni minime relative alle imprese da contattare e quali requisiti tecnici deve possedere l'elenco anagrafico da acquistare;
- attraverso quali elaborazioni le imprese dell'elenco vanno ripartite tra gli agenti di vendita, ciascuno operante su una o più province del territorio italiano;
- su quali parametri (per esempio, numero di imprese effettivamente contattate, numero di contratti conclusi o fatturato realizzato per agente) e con quali procedure verrà valutata "a posteriori" l'efficacia dell'intera operazione.

Sviluppare anche una parte significativa di uno degli ultimi due punti.

19 ..... *Esame di Stato dell'ITIS a indirizzo informatico - Corso di ordinamento - 1991*
Prova scritta di Informatica

Registro di classe

Il registro di Classe ha, per ogni giorno dell'anno scolastico, una zona in cui riportare, ora per ora, la Materia di effettivo insegnamento e la Firma (o più firme nelle ore di compresenza) degli Insegnanti effettivamente presenti in classe a quell'ora.
In un'altra zona si annotano i nomi degli Alunni assenti e di quelli che presentano permessi di entrata e/o uscita per ore intermedie. Vi è poi spazio per altre note.
Si pensa di poter comunque rilevare le ore che non sono dedicate all'insegnamento, ma sono occupate da "materie" come "Uscita anticipata", "Assemblea", "Supplenza", e simili.

Si deve istituire un sistema informatico che, raccogliendo i dati dei diari di tutte le classi di una scuola, consenta almeno le seguenti operazioni:

1. A fine giornata più impiegati aggiornano contemporaneamente gli archivi, ciascuno dal proprio posto di lavoro.
2. A richiesta di un dato Consiglio di Classe si stampa un bollettino che riporta, per quella classe:
   a. il numero delle ore di lezione effettivamente svolte per ciascuna disciplina;
   b. il numero delle ore di assenza per ogni alunno in ciascuna disciplina.
3. A richiesta della presidenza si stampa un bollettino che riporta:
   a. per ogni insegnante il numero di giorni di assenza;
   b. per ogni classe il numero di giorni di assenza di ciascun alunno e il numero medio di giorni di assenza degli alunni.

Loescher Editore - Vietata la vendita e la diffusione

ESEMPI DI SVOLGIMENTO 14

14.1 OLIMPIADI DI INFORMATICA

In occasioni delle Olimpiadi Internazionali di Informatica 2008, la società organizzatrice desidera realizzare un sistema informatico per la gestione delle gare e degli "atleti".

La base di dati deve consentire la memorizzazione delle informazioni

- degli atleti, che possono partecipare alle gare sia singolarmente sia raggruppati in squadre
- delle gare nelle varie fasi
- delle sedi di gara

Le Olimpiadi prevedono una fase scolastica (in ciascun istituto scolastico partecipante), una fase regionale, una finale nazionale e la gara internazionale che designerà il vincitore e la squadra vincitrice.

Il candidato, fatte le opportune **ipotesi aggiuntive**, realizzi:

1. un'**analisi della realtà di riferimento** individuando le possibili soluzioni e scelga quella che a suo motivato giudizio è la più idonea a rispondere alle specifiche indicate

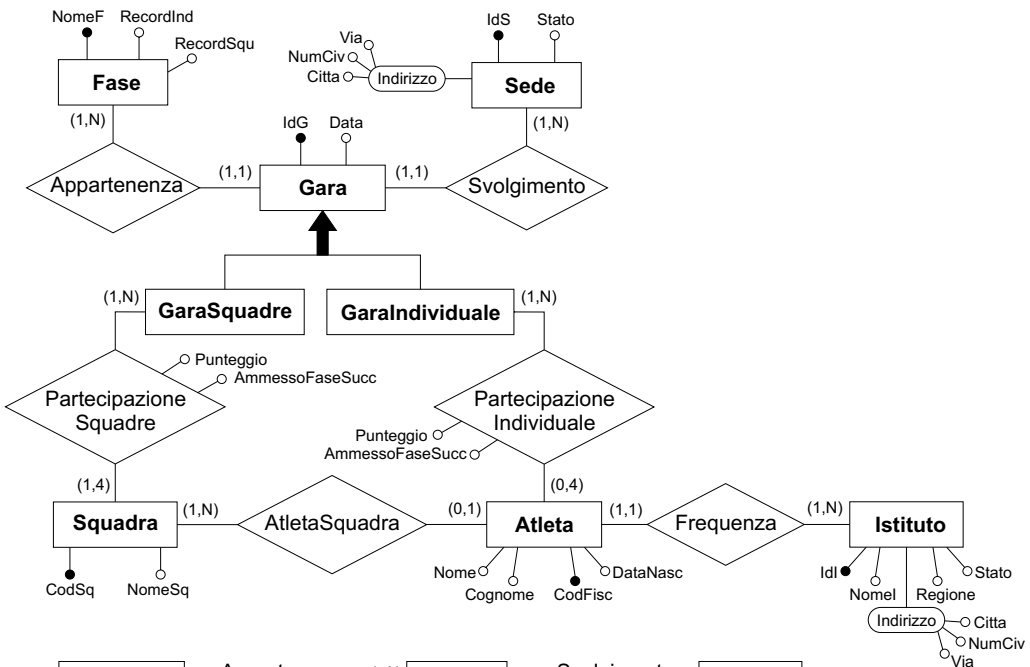
Solo gli studenti e le squadre che hanno vinto una certa fase possono partecipare alle gare della fase successiva.
In tutte le fasi una squadra è composta dallo stesso gruppo di atleti, tutti dello stesso istituto.
Le gare dalla fase successiva si svolgono solo quando sono finite tutte le gare della fase precedente.
Le gare per le squadre sono indipendenti dalle gare individuali.

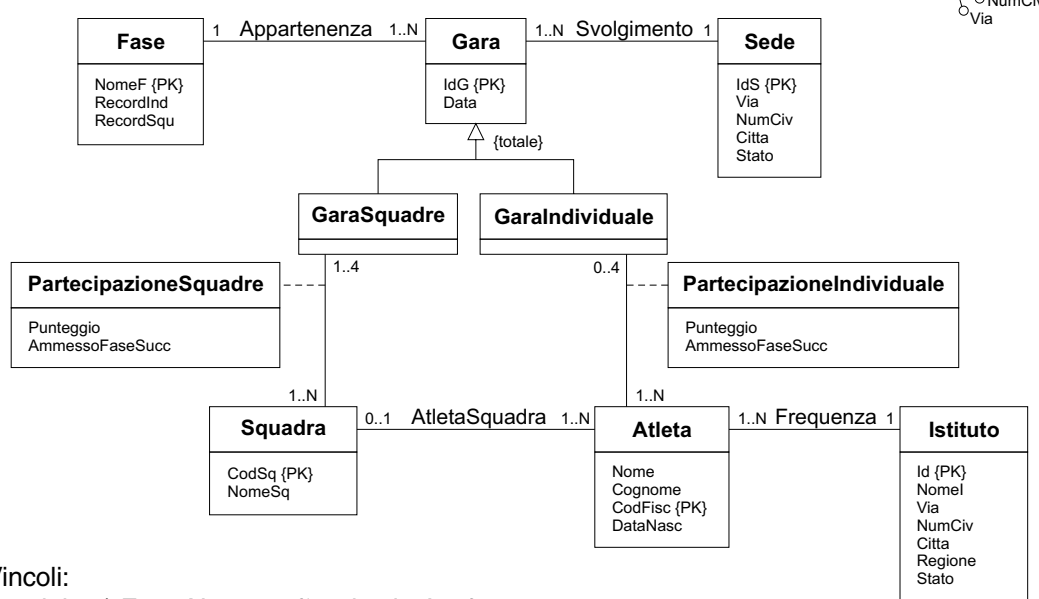
Per ogni **istituto** vengono memorizzati nome, indirizzo (via, numero civico, città), regione e stato. Inoltre ogni istituto ha un identificatore che viene usato per l'inserimento degli atleti.
Per gli **atleti** vengono memorizzati nome, cognome, data di nascita, codice fiscale (usato come identificatore) e l'istituto scolastico che frequenta. Possono esserci atleti che partecipano solo come membri di una squadra e che non partecipano a nessuna gara individuale.
Ogni atleta può far parte al più di una squadra.
Le **squadre** hanno un nome e un identificatore, possono essere formate solo da studenti dello stesso istituto e rimangono le stesse per tutte le fasi delle olimpiadi.
Per ogni **sede** in cui si svolgeranno delle gare sono memorizzati l'indirizzo (via, numero, città, stato) e un identificatore della sede.

Loescher Editore - Vietata la vendita e la diffusione

Le **gare** sono caratterizzate da fase, sede, data e identificatore. Ci saranno gare alle quali non parteciperà nessun atleta individualmente (cioè gare riservate alle squadre) e gare riservate ai singoli atleti in cui le squadre non saranno ammesse.
Le **fasi** sono prestabilite, quindi i loro nomi sono di tipo enumerativo.

2. uno **schema concettuale** della base di dati





Vincoli:
Dominio.1) Fase.Nome := {‘scolastica’, ...}
Dominio.2) GareIndividuali.AmmessoFaseSucc := {'sì', 'no')} (per comodità di stampa risultato)
Dominio.3) GareSquadre.AmmessoFaseSucc := {'sì', 'no')} (per comodità di stampa risultato)
RV.1) Un atleta non può partecipare a due gare relative alla stessa fase.
RV.2) Una squadra non può partecipare a due gare relative alla stessa fase.
RV.3) Un atleta deve essere stato ammesso alla fase a cui partecipa.

Loescher Editore - Vietata la vendita e la diffusione

3. uno **schema logico** della base di dati

Istituti (IdI, NomeI, Via, NumCiv, Citta, Regione, Stato)
Atleti (CodFisc, Nome, Cognome, DataNasc, IdIstituto)
Squadre (CodSq, NomeSq)
Sedi (IdS, Via, NumCiv, Citta, Stato)
Fasi (NomeF, RecordInd, RecordSqu)
Gare (IdG, Fase, IdSede, Data)
GareIndividuali (IdAtleta, IdGara, Punteggio, AmmessoFaseSucc)
GareSquadre (IdSquadra, IdGara, Punteggio, AmmessoFaseSucc)
AtletaSquadra (IdAtleta, IdSquadra)

Atleti.IdIstituto è in integrità referenziale con *Istituti.IdI*;
Gare.Fase è in integrità referenziale con *Fasi.NomeF*;
Gare.IdSede è in integrità referenziale con *Sedi.IdS*;
GareIndividuali.IdAtleta è in integrità referenziale con *Atleti.CodFisc*;
GareIndividuali.IdGara è in integrità referenziale con *Gare.IdG*;
GareSquadre.IdSquadra è in integrità referenziale con *Squadre.CodSq*;
GareSquadre.IdGara è in integrità referenziale con *Gare.IdG*;
AtletaSquadra.IdAtleta è in integrità referenziale con *Atleti.CodFisc*;
AtletaSquadra.IdSquadra è in integrità referenziale con *Squadre.CodSq*.

4. la **definizione delle relazioni** della base di dati in linguaggio **SQL**

Soluzione in MySQL

```
create table Istituti (
    IdI      char(5) primary key,
    NomeI    varchar(20) not null,
    Via      varchar(20),
    NumCiv   integer,
    Citta    varchar(20) not null,
    Regione  varchar(20) not null,
    Stato varchar(20) not null
)ENGINE=innodb;

create table Atleti (
    CodFisc     char(16) primary key,
    Nome        varchar(20) not null,
    Cognome     varchar(20) not null,
    DataNasc    date not null,
    IdIstituto  char(5) not null,
    foreign key (IdIstituto) references Istituti(IdI)
)ENGINE=innodb;

create table Squadre (
    CodSq   char(5) primary key,
    NomeSq  varchar(20) not null
)ENGINE=innodb;

create table Sedi (
    IdS     char(5) primary key,
```

Loescher Editore - Vietata la vendita e la diffusione

```
    Via     varchar(20) not null,
    NumCiv  integer not null,
    Citta   varchar(20) not null,
    Stato   varchar(20) not null
)ENGINE=innodb;

create table Fasi (
    NomeF
enum('scolastica','regionale','nazionale','internazionale')
primary key,
    RecordInd  integer unsigned not null default 0,
    RecordSqu  integer unsigned not null default 0
)ENGINE=innodb;

create table Gare (
    IdG     char(5) primary key,
    Fase
enum('scolastica','regionale','nazionale','internazionale') not
null,
    IdSede  char(5) not null,
    Data    date,
    foreign key (Fase) references Fasi(NomeF),
    foreign key (IdSede) references Sedi(IdS)
)ENGINE=innodb;

create table GareIndividuali (
    IdAtleta         char(16) not null,
    IdGara           char(5) not null,
    Punteggio        integer default NULL,
    AmmessoFaseSucc  enum ('sì', 'no'),
    primary key(IdAtleta,IdGara),
    foreign key (IdAtleta) references Atleti(CodFisc),
    foreign key (IdGara) references Gare(IdG)
)ENGINE=innodb;

create table GareSquadre (
    IdSquadra        char(5) not null,
    IdGara           char(5) not null,
    Punteggio        integer default NULL,
    AmmessoFaseSucc  enum ('sì', 'no'),
    primary key (IdSquadra,IdGara),
    foreign key (IdSquadra) references Squadre(CodSq),
    foreign key (IdGara) references Gare(IdG)
)ENGINE=innodb;

create table AtletaSquadra (
    IdAtleta   char(16) primary key,
    IdSquadra  char(5) not null,
    foreign key (IdSquadra) references Squadre(CodSq),
    foreign key (IdAtleta) references Atleti(CodFisc)
)ENGINE=innodb;
```

Loescher Editore - Vietata la vendita e la diffusione

5. le seguenti **interrogazioni** espresse in linguaggio **SQL**:

- stampare l'elenco degli atleti raggruppati per squadre per ogni singola fase

```
SELECT G.Fase, S.NomeSq, A.Cognome, A.Nome
FROM Atleti AS A, AtletaSquadra AS A_S, Squadre AS S, GareSquadre
AS G_S, Gare AS G
WHERE (A.CodFisc=A_S.IdAtleta) AND (A_S.IdSquadra=S.CodSq) AND
      (S.CodSq=G_S.IdSquadra) AND (G_S.IdGara=G.IdG)
ORDER BY G.Fase, S.NomeSq, A.Cognome, A.Nome;
```

- dato il nome di un atleta stampare i risultati ottenuti nelle diverse gare alle quali ha partecipato

```
SELECT GI.IdGara, GI.Punteggio, GI.AmmessoFaseSucc
FROM Atleti AS A, GareIndividuali AS GI
WHERE (A.CodFisc=GI.IdAtleta) AND
      (A.Cognome=@cognome) AND (A.Nome=@nome)
UNION
SELECT GS.IdGara, GS.Punteggio, GS.AmmessoFaseSucc
FROM Atleti AS A, AtletaSquadra AS AtS, GareSquadre AS GS
WHERE (A.CodFisc=AtS.IdAtleta) AND (AtS.IdSquadra=GS.IdSquadra) AND
      (A.Cognome=@cognome) AND (A.Nome=@nome);
```

- stampare il calendario delle gare

```
SELECT G.Data, G.Fase, G.IdG, S.Via, S.NumCiv, S.Citta, S.Stato
FROM Gare AS G JOIN Sedi AS S ON (G.IdSede=S.IdS)
ORDER BY G.Data;
```

- stampare una scheda informativa (cognome, nome, istituto scolastico di provenienza, nazionalità) del vincitore e della squadra vincitrice

Vincitore

```
SELECT A.Nome, A.Cognome, I.NomeI, I.Stato
FROM Istituti AS I, Atleti AS A, GareIndividuali AS GI, Gare AS G
WHERE (I.IdI=A.IdIstituto) AND (A.CodFisc=GI.IdAtleta) AND
      (GI.IdGara=G.IdG) AND (GI.AmmessoFaseSucc='sì') AND
      (G.Fase='internazionale');
```

Squadra vincitrice

```
SELECT S.NomeSq, A.Nome, A.Cognome, I.NomeI, I.Stato
FROM Istituti AS I, Atleti AS A, AtletaSquadra AS A_S, Squadre AS
S, GareSquadre AS GS, Gare AS G
WHERE (I.IdI=A.IdIstituto) AND (A.CodFisc=A_S.IdAtleta) AND
      (A_S.IdSquadra=S.CodSq) AND (S.CodSq=GS.IdSquadra) AND
      (GS.IdGara=G.IdG) AND (GS.AmmessoFaseSucc='sì') AND
      (G.Fase='internazionale');
```

- stampare la classifica per ciascuna gara (a parità di punteggio vengono privilegiati gli atleti più giovani)

Loescher Editore - Vietata la vendita e la diffusione

Esito delle gare individuali

```
SELECT G.IdG, G.Fase, A.Cognome, A.Nome, GI.Punteggio, A.DataNasc
FROM Gare AS G, GareIndividuali AS GI, Atleti AS A
WHERE (G.IdG=GI.IdGara) AND (GI.IdAtleta=A.CodFisc)
ORDER BY G.Fase, G.IdG, GI.Punteggio DESC, A.DataNasc DESC;
```

Esito delle gare a squadre

```
SELECT G.IdG, G.Fase, S.NomeSq, GS.Punteggio
FROM Gare AS G, GareSquadre AS GS, Squadre AS S
WHERE (G.IdG=GS.IdGara) AND (GS.IdSquadra=S.codSq)
ORDER BY G.Fase, G.IdG, GS.Punteggio DESC;
```

- aggiornare, per ciascuna fase (scolastica-regionale-nazionale-internazionale) gli eventuali punteggi record

Modifica del record individuale relativo alla fase scolastica

```
UPDATE Fasi SET RecordInd=(
      SELECT MAX(GI.Punteggio)
      FROM GareIndividuali AS GI JOIN Gare AS G ON (GI.IdGara=G.IdG)
      WHERE G.Fase='scolastica')
WHERE NomeF='scolastica' AND RecordInd<(
      SELECT MAX(GI.Punteggio)
      FROM GareIndividuali AS GI JOIN Gare AS G ON (GI.IdGara=G.IdG)
      WHERE G.Fase='scolastica');
```

Soluzione con una vista

```
CREATE VIEW Ris (P) AS
      (SELECT RecordInd
      FROM Fasi
      WHERE NomeF='scolastica')
      UNION
      (SELECT MAX(GI.Punteggio)
      FROM GareIndividuali AS GI JOIN Gare AS G ON (GI.IdGara=G.IdG)
      WHERE G.Fase='scolastica');

UPDATE Fasi SET RecordInd=(
      SELECT MAX(P)
      FROM Ris)
WHERE NomeF='scolastica';

DROP VIEW Ris;
```

Nota

Per le altre fasi si procede in modo analogo.

- calcolare il punteggio medio ottenuto durante la prima selezione, per ciascun istituto scolastico

```
SELECT I.NomeI, avg(GI.Punteggio) AS Media
FROM Gare AS G, GareIndividuali AS GI, Atleti AS A, Istituti AS I
WHERE (G.IdG=GI.IdGara) AND (GI.IdAtleta=A.CodFisc) AND
(A.IdIstituto=I.IdI) AND (G.Fase='scolastica')
GROUP BY I.NomeI
ORDER BY I.NomeI;
```

Loescher Editore - Vietata la vendita e la diffusione

- stampare per ciascuna squadra il numero di "atleti" partecipanti e l'età media

```
SELECT S.CodSq, S.NomeSq, count(A_S.IdSquadra) AS NumPartecipanti,
     AVG((YEAR(CURDATE())-YEAR(A.DataNasc)) -
         (RIGHT(CURDATE(),5)<RIGHT(A.DataNasc,5)))
     AS etaMedia
FROM Squadre AS S, AtletaSquadra AS A_S, Atleti AS A
WHERE (S.CodSq=A_S.IdSquadra) AND (A_S.IdAtleta=A.CodFisc)
GROUP BY A_S.IdSquadra;
```

l'età effettiva dell'atleta viene calcolata considerando se ha già compiuto gli anni o no (fino al giorno prima del compleanno).

6. l'**interfaccia utente** che il candidato intende proporre per interagire con la base di dati e codificare in un **linguaggio di programmazione a scelta** un segmento significativo del progetto realizzato.

Tutta l'interfaccia utente per interagire col database può essere realizzata mediante pagine Web.

Le funzioni disponibili riguardano l'inserimento, la modifica e la cancellazione dei dati e vari tipi di visualizzazioni.

Il sito viene realizzato in PHP.

I segmenti significativi scelti sono l'inserimento e la modifica dei dati di un atleta che possono essere gestiti da un'unica pagina PHP e la visualizzazione dei dati degli atleti, che può essere usata anche per scegliere un atleta di cui si vogliono modificare o cancellare i dati.

InsModAtleta.php

```
<html>
<head><title>Atleti iscritti</title></head>
<?php
      include("mysql.php");
      connetti();
?>
<body>
<div style="text-align:center">
<h2>Olimpiadi di informatica 2008/2009</h2>
<?php
      if(isset($_POST['CodFisc'])){
```

```
        if($_POST['azione']=="mod"){ // modifica
            $query="UPDATE Atleti SET
                CodFisc='".$_POST['CodFisc']."',
                Nome='".$_POST['Nome']."',
                Cognome='".$_POST['Cognome']."',
                DataNasc='".$_POST['DataNasc']."',
                IdIstituto='".$_POST['Istituti']."'
                WHERE CodFisc='".$_POST['CodFiscOld']."';";
            $esito="Modifica avvenuta con successo.";
        }else{ // inserimento
            $query="INSERT INTO Atleti (Nome, Cognome, DataNasc, CodFisc, IdIstituto) VALUES
('".$_POST['Nome']."','".$_POST['Cognome']."','".$_POST['DataNasc']."','".$_POST['CodFisc']."','".$_POST['Istituti']."');";
            $esito="Inserimento avvenuto con successo.";
        }// if-else
        if(mysql_query($query))
            echo $esito;
        else
            echo "Si e' verificato un errore";
        echo "<br/>";
    } else {
        $nome=$cognome=$dataNasc=$codFis="";
        if(isset($_GET['id'])){
            $query = "SELECT *  FROM Atleti  WHERE CodFisc='".$_GET['id']."'";
            $result= mysql_query($query);
            $row = mysql_fetch_array($result);
            $nome=$row['Nome'];
            $cognome=$row['Cognome'];
            $dataNasc=$row['DataNasc'];
            $codFis=$row['CodFisc'];
            echo "<h3>Modifica Atleta</h3>";
        }else
            echo "<h3>Inserisci Atleta</h3>";
?>
        </div>
        <form action="InsModAtleta.php" method="post">
            <table>
                <tr>
                    <td style="text-align:right">Nome:</td>
                    <td><input type="text" name="Nome" value="<?php echo $nome; ?>"/></td>
                </tr>
                <tr>
                    <td style="text-align:right">Cognome:</td>
                    <td><input type="text" name="Cognome" value="<?php echo $cognome; ?>"/></td>
                </tr>
                <tr>
                    <td style="text-align:right">Data di nascita:</td>
                    <td><input type="text" name="DataNasc" value="<?php echo $dataNasc; ?>"/></td>
                </tr>
                <tr>
                    <td style="text-align:right">Codice Fiscale:</td>
                    <td><input type="text" name="CodFisc" value="<?php echo $codFis; ?>"/></td>
                </tr>
```

Loescher Editore - Vietata la vendita e la diffusione

```
                <tr>
                    <td>Seleziona l'istituto:</td>
                    <td><select name="Istituti">
                    <?php
                        $query = "SELECT NomeI, Citta, IdI  FROM Istituti ORDER BY NomeI";
                        $result= mysql_query($query);
                        while ($row = mysql_fetch_array($result)){
                        echo("<option value=\"".$row['IdI']."\">".$row['NomeI']." (".$row['Citta']")</option>");
                        }//while
                    ?>
                    </select></td>
                </tr>
            </table>
            <?php
                if(isset($_GET['id']))
                    echo "<input type=\"hidden\" name=\"CodFiscOld\" value=\"".$codFis."\"/>";
            ?>
            <input type="hidden" name="azione" value="<?php echo(isset($_GET['id'])?"mod":"ins"); ?>"/>
            <br/><input type="submit" value="Invia">
        </form>
<?php
    }// if-else
?>
<p><a href="index.php">Back</a></p>
</body>
</html>
```

visAtleti.php

Olimpiadi di informatica 2008/2009

Visualizzazione di tutti gli atleti iscritti

Cognome	Nome	Data di nascita	Codice Fiscale	Istituto	Citta'	Regione	
XXXX	XX	XXXXXXXX	XXXXXX	XXXXXX	XXXX	XXXX	Modifica
XXXX	XX	XXXXXXXX	XXXXXX	XXXXXX	XXXX	XXXX	Modifica

```
<html>
<head><title>Atleti iscritti</title>
</head>
<?php
    include("mysql.php");
    connetti();
?>
<body>
<div style="text-align:center"><h2>Olimpiadi di informatica 2008/2009</h2>
<h3>Visualizzazione di tutti gli atleti iscritti</h3>
</div>
<table cellpadding="3" border="1">
    <tr>
        <td><b>Cognome</b></td>
        <td><b>Nome</b></td>
        <td><b>Data di nascita</b></td>
        <td><b>Codice Fiscale</b></td>
        <td><b>Istituto</b></td>
        <td><b>Citta'</b></td>
        <td><b>Regione</b></td>
        <?php
            if(isset($_GET['op']))
                echo "<td > </td>";
        ?>
```

```
        </tr>
<?php
        $query = "SELECT A.Nome, A.Cognome, A.DataNasc, A.CodFisc, I.NomeI, I.Citta, I.Regione
                FROM Atleti AS A JOIN Istituti AS I ON (A.idIstituto=I.IdI)
                ORDER BY A.Cognome, A.Nome";
        $result= mysql_query($query);
        while ($row = mysql_fetch_array($result)){
            echo("<tr>
                <td style=\"text-align:center\">".$row['Cognome']."</td>
                <td style=\"text-align:center\">".$row['Nome']."</td>
                <td style=\"text-align:center\">".$row['DataNasc']."</td>
                <td style=\"text-align:center\">".$row['CodFisc']."</td>
                <td style=\"text-align:center\">".$row['NomeI']."</td>
                <td style=\"text-align:center\">".$row['Citta']."</td>
                <td style=\"text-align:center\">".$row['Regione']."</td>");
                if(isset($_GET['op'])){
                    if($_GET['op']=="canc"){
                        $url="cancAtleta.php";
                        $azione="Cancella";
                    }else{
                        $url="InsModAtleta.php";
                        $azione="Modifica";
                    }//if-else
                    echo ("<td><a href=\"".$url."?id=".$row['CodFisc']."\">".$azione."</a></td>");
                }//if
            echo "</tr>";
        }//while
?>
</table>
<p><a href="index.php">Back</a></p>
</body>
</html>
```

7. un **sito Internet** che presenti al pubblico le classifiche delle diverse gare.

La pagina principale presenta l'elenco delle gare svolte; scegliendo una gara si ottiene la classifica relativa.

index.php

```
<html>
<head><title>Classifica delle Gare</title></head>
<?php
        include("../mysql.php");
        connetti();
?>
<body>
<div style="text-align:center">
<h2>Olimpiadi di informatica 2008/2009</h2>
<h3> Classifica delle Gare finora svolte</h3>
</div>
<?php
        function stampaGare($query){
```

Loescher Editore - Vietata la vendita e la diffusione

```
            $result= mysql_query($query);
            while ($row = mysql_fetch_array($result)){
                echo ("<li><a href=\"risultati.php?idg=".$row['IdG']."&tipo=".$row['Tipo']."\">
                Gara del ".$row['Data']." svoltasi in via ".$row['Via']." ".$row['NumCiv']., ".$row['Citta'].
                ", ".$row['Stato']."</a></li>");
            }//while
                }//stampaGare
?>
<b><p>Fase Scolastica:</p></b><i>Gare individuali:</i>
<ul>
<?php
    $queryGIS = "SELECT DISTINCT G.IdG, G.Data, S.Via, S.NumCiv, S.Citta, S.Stato
        FROM Gare AS G, Sedi AS S, GareIndividuali AS GI WHERE G.IdSede=S.IdS
        AND G.IdG=GI.IdGara AND G.Fase='scolastica' AND G.Data<CURDATE()
        ORDER BY Data";
    stampaGare($queryGIS,"i");
?>
</ul>
<i>Gare a squadre:</i>
<ul>
<?php
    $queryGSS = "SELECT DISTINCT G.IdG, G.Data, S.Via, S.NumCiv, S.Citta, S.Stato
        FROM Gare AS G, Sedi AS S, GareSquadre AS GS WHERE G.IdSede=S.IdS
        AND G.IdG=GS.IdGara AND G.Fase='scolastica' AND G.Data<CURDATE()
        ORDER BY Data";
    stampaGare($queryGSS,"s");
?>
</ul>
<b><p>Fase Regionale:</p></b><i>Gare individuali:</i>
<ul>
<?php
    $queryGIR = "SELECT DISTINCT G.IdG, G.Data, S.Via, S.NumCiv, S.Citta, S.Stato
        FROM Gare AS G, Sedi AS S, GareIndividuali AS GI WHERE G.IdSede=S.IdS
        AND G.IdG=GI.IdGara AND G.Fase='regionale' AND G.Data<CURDATE()
        ORDER BY Data";
    stampaGare($queryGIR,"i");
?>
</ul>
<i>Gare a squadre:</i>
<ul>
<?php
    $queryGSR = "SELECT DISTINCT G.IdG, G.Data, S.Via, S.NumCiv, S.Citta, S.Stato
        FROM Gare AS G, Sedi AS S, GareSquadre AS GS WHERE G.IdSede=S.IdS
        AND G.IdG=GS.IdGara AND G.Fase='regionale' AND G.Data<CURDATE()
        ORDER BY Data";
    stampaGare($queryGSR,"s");
?>
</ul>
<b><p>Fase Nazionale:</p></b><i>Gare individuali:</i>
<ul>
<?php
```

Loescher Editore - Vietata la vendita e la diffusione

```
        $queryGIN = "SELECT DISTINCT G.IdG, G.Data, S.Via, S.NumCiv, S.Citta, S.Stato
            FROM Gare AS G, Sedi AS S, GareIndividuali AS GI
            WHERE G.IdSede=S.IdS AND G.IdG=GI.IdGara AND G.Fase='nazionale'
            AND G.Data<CURDATE()
            ORDER BY Data";
        stampaGare($queryGIN,"i");
    ?>
    </ul>
    <i>Gare a squadre:</i>
    <ul>
    <?php
        $queryGSN = "SELECT DISTINCT G.IdG, G.Data, S.Via, S.NumCiv, S.Citta, S.Stato
            FROM Gare AS G, Sedi AS S, GareSquadre AS GS
            WHERE G.IdSede=S.IdS AND G.IdG=GS.IdGara AND G.Fase='nazionale'
            AND G.Data<CURDATE()
            ORDER BY Data";
        stampaGare($queryGSN,"s");
    ?>
    </ul>
    <b><p>Fase Internazionale:</p></b><i>Gare individuali:</i>
    <ul>
    <?php
        $queryGII = "SELECT DISTINCT G.IdG, G.Data, S.Via, S.NumCiv, S.Citta, S.Stato
            FROM Gare AS G, Sedi AS S, GareIndividuali AS GI WHERE G.IdSede=S.IdS
            AND G.IdG=GI.IdGara AND G.Fase='internazionale' AND G.Data<CURDATE()
            ORDER BY Data";
        stampaGare($queryGII,"i");
    ?>
    </ul>
    <i>Gare a squadre:</i>
    <ul>
    <?php
        $queryGSI = "SELECT DISTINCT G.IdG, G.Data, S.Via, S.NumCiv, S.Citta, S.Stato
            FROM Gare AS G, Sedi AS S, GareSquadre AS GS
            WHERE G.IdSede=S.IdS AND G.IdG=GS.IdGara AND G.Fase='internazionale'
            AND G.Data<CURDATE()
            ORDER BY Data";
        stampaGare($queryGSI,"s");
    ?>
    </ul>
    </body>
    </html>
```

risultati.php

```
    <html>
    <head><title>Classifica</title></head>
    <?php
    include("../mysql.php");
    connetti();
    ?>
    <body>
    <div style="text-align:center">
```

Olimpiadi di informatica 2008/2009

Classifica della Gara del 2008-09-20 svoltasi in via XXXX , Milano, Italia

Atleta	Punteggio	Fase Successiva
XXX	XXX	XXX
XXX	XXX	XXX

torna alle Gare

Loescher Editore - Vietata la vendita e la diffusione

```
<h2>Olimpiadi di informatica 2008/2009</h2>
<h3> Classifica della Gara
<?php
  $id=$_GET['idg'];
      $queryGara = "SELECT *
            FROM Gare AS G JOIN Sedi AS S ON (G.IdSede=S.IdS)
            WHERE G.IdG='$id'";
      $result= mysql_query($queryGara);
      while ($row = mysql_fetch_array($result)){
            echo(" del ".$row['Data']." svoltasi in via ".$row['Via']." ".$row['NumCiv']."., ".$row['Citta']."., ".$row['Stato']);
      }//while
?>
</h3>
</div>
<table cellpadding="3" border="1">
<?php
      if($_GET['tipo']=='i'){
            echo "<tr><td><b>Atleta</b></td><td><b>Punteggio</b></td><td><b>Fase Successiva</b>
                  </td></tr>";
            $queryClass = "SELECT *
                  FROM GareIndividuali AS GI JOIN Atleti AS A ON (A.CodFisc=GI.IdAtleta)
                  WHERE GI.IdGara='$id'
                  ORDER BY GI.Punteggio DESC, A.DataNasc DESC";
            $result= mysql_query($queryClass);
            while ($row = mysql_fetch_array($result)){
                  echo("<tr>
                        <td>".$row['Cognome']." ".$row['Nome']."</td>
                        <td style=\"text-align:center\">".$row['Punteggio']."</td>
                        <td style=\"text-align:center\">".$row['AmmessoFaseSucc']."</td>
                        </tr>");
            }//while
      } else {
            echo "<tr><td><b>Squadra</b></td><td><b>Punteggio</b></td><td><b>Fase Successiva
                  </b></td></tr>";
            $queryClass = "SELECT *
                  FROM GareSquadre AS GS JOIN Squadre AS S ON (S.CodSq=GS.IdSquadra)
                  WHERE GS.IdGara='$id'
                  ORDER BY GS.Punteggio DESC";
            $result= mysql_query($queryClass);
            while ($row = mysql_fetch_array($result)){
                  echo("<tr>
                        <td>$row[NomeSq]</td>
                        <td style=\"text-align:center;\">".$row['Punteggio']."</td>
                        <td style=\"text-align:center;\">".$row['AmmessoFaseSucc']."</td>
                        </tr>");
            }//while
      }//if-else
?>
</table>
<hr/><a href="index.php">torna alle Gare</a>
</body>
</html>
```

14.2 TRATTAMENTO DI DATI SENSIBILI

Il Codice della Privacy prevede una serie di adempimenti per i soggetti pubblici e privati che trattano dati sensibili con supporti informatici. Il candidato illustri le possibili procedure per:

- garantire l'accesso ai documenti soltanto a soggetti autorizzati;

Il trattamento dei dati personali è disciplinato dal **Decreto legislativo 196 del 30 giugno 2003** "Codice in materia di protezione dei dati personali".
In particolare "i dati sensibili possono essere oggetto di trattamento solo con il consenso scritto dell'interessato e previa autorizzazione del Garante...".
Il Decreto stabilisce le misure di sicurezza che devono essere adottate per il trattamento dei dati con strumenti elettronici e che comprendono tra l'altro l'impiego di metodi di autenticazione e autorizzazione.
L'**autenticazione** permette di verificare l'identità di chi accede al computer e quindi di determinare chi è l'autore di una operazione.

Ci sono tre modi fondamentali per **verificare l'identità** di un utente: mediante una **password** (cioè qualcosa che solo il soggetto sa), mediante un **token**, cioè una smart card (carta magnetica personalizzate) o una chiave USB (cioè qualcosa che solo il soggetto ha), e mediante **riconoscimento biometrico** di caratteristiche fisiche come le impronte digitali o la scansione della retina o l'analisi della voce (cioè qualcosa che solo il soggetto è).
L'utilizzo di una **password** è il modo più comune, perché è semplice e poco costoso, ma non è molto sicuro.
La password può essere facilmente carpita all'utente, anche solo standogli vicino mentre la digita (**shoulder surfing**), può essere indovinata per tentativi (**guessing**) o provando in modo esaustivo tutti i valori possibili (**brute forcing**), per esempio usando i termini di un dizionario (**dictionary attack**). E possibile anche intercettare le informazioni che viaggiano sulla rete (**sniffing**), o ottenerle direttamente all'utente con tecniche di **social engineering** (chiedendole direttamente, in modo apparentemente casuale), **phishing** (richiesta di informazioni all'utente tramite e-mail spacciandosi per qualcun altro che ne avrebbe diritto) o uso di **spyware** (software che raccoglie informazioni sull'utente e le invia attraverso Internet).

Per difendersi è opportuno usare sistemi di autenticazione che non facciano transitare la password sulla rete o metodi di crittografia, in modo che le password non siano mai in chiaro, suggerire agli utenti di usare una password non banale e spiegare che non devono scriverla e non devono comunicarla a nessuno.

Dopo aver effettuato l'autenticazione il **sistema di protezione** può effettuare il controllo degli accessi garantendo che l'utente possa compiere solo le operazioni **autorizzate** in base alla sua identità. Le operazioni effettuate possono essere controllare mediante l'**auditing degli eventi**.

Usando tecniche di autenticazione e autorizzazione si può controllare chi può accedere ai documenti per utilizzarli o anche solo per effettuarne il backup.

Le copie dei documenti devono utilizzare **tecniche crittografiche** ed essere protette con la stessa cura dei documenti originali.

- gestire i *backup* periodici dei soli documenti contenenti dati sensibili;
- simulare periodicamente la perdita dei dati e il recupero degli stessi dalle copie di *backup*;

Il **backup** consiste nella copia di informazioni su un altro supporto (unità usate comunemente sono nastri, CDROM e DVD, un altro hard disk o l'hard disk di un altro computer in rete).

Loescher Editore - Vietata la vendita e la diffusione

Si può eseguire il backup di un set di cartelle o file selezionati. Una **politica di backup** può consistere nell'effettuare un backup intero una volta al mese e un backup incrementale ogni giorno. Conviene inoltre effettuare il backup alternando l'uso di due insiemi di supporti diversi (in modo da mantenere anche le copie di backup del mese precedente).
Si possono **ripristinare** i dati dalle copie di backup; se sono stati effettuati dei backup incrementali bisogna ripristinare prima il backup completo e poi uno alla volta i vari backup incrementali.

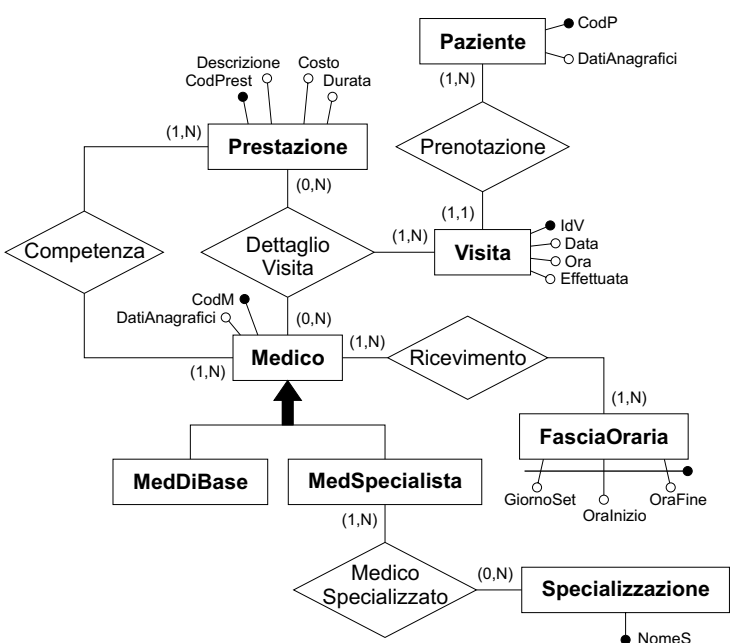
- modificare periodicamente i parametri di accesso ai documenti protetti.

Si possono utilizzare **meccanismi automatici** che costringano gli utenti a **cambiare periodicamente la password**.
In Windows l'amministratore di sistema può impostare dei criteri per l'uso delle password e dei criteri di protezione per gli account da *Strumenti di amministrazione*, *Criteri di protezione locali*.
Oltre a impostare una validità massima per la password è possibile anche imporre una lunghezza minima e l'unicità della password in modo che non si possano riutilizzare password già usate in precedenza. È inoltre possibile **bloccare l'account** in caso di tentativi di collegamento con password errate.

Successivamente sviluppi il seguente punto.

Uno studio medico associato, in cui operano medici di base e medici specialisti, dà incarico a una società d'informatica di progettare e realizzare un database contenente i dati anagrafici e professionali dei medici, l'orario delle visite, i tempi medi previsti per ogni visita, il costo delle singole prestazioni, i dati anagrafici dei pazienti e le prestazioni richieste.

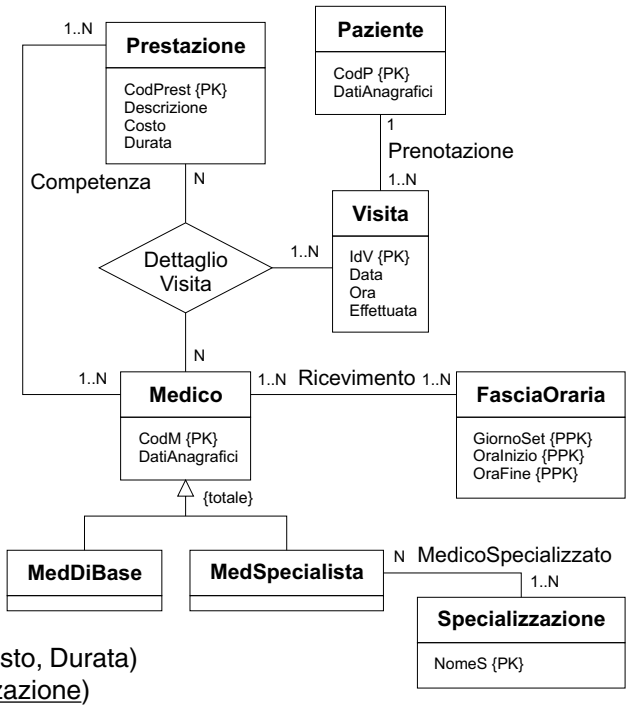
Ipotesi aggiuntive:
1) Essendoci medici specialisti e medici di base, non tutti i medici possono effettuare tutte le prestazioni. A specializzazioni differenti corrispondono prestazioni diverse.



Vincoli:
1) Un medico durante una visita può effettuare solo prestazioni che rientrano nelle sue competenze, e quindi che compaiono nella relazione *Competenze*.
2) L'orario di una visita deve essere compatibile con l'orario di ricevimento del medico prenotato.

Dominio.1) FasciaOraria.GiornoSet := {'lunedì', ... }

Loescher Editore - Vietata la vendita e la diffusione



Schema relazionale

Pazienti (CodP, DatiAnagrafici)
Medici (CodM, DatiAnagrafici)
Competenza (Medico, Prestazione)
Specializzazioni (NomeS)
Prestazioni (CodPrest, Descrizione, Costo, Durata)
MediciSpecializzati (Medico, Specializzazione)
FasceOrarie (GiornoSet, OraInizio, OraFine)
Ricevimento (CodM, GiornoSet, OraInizio, OraFine)
Visite (IdV, Paziente, Medico, Data, Ora, Effettuata)
DettaglioVisita (Visita, Prestazione)

Competenza.Medico è in integrità referenziale con *Medici.CodM*;
Competenza.Prestazione è in integrità referenziale con *Prestazioni.CodPrest*;
MediciSpecializzati.Medico è in integrità referenziale con *Medici.CodM*;
MediciSpecializzati.Specializzazione è in integrità referenziale con *Specializzazioni.NomeS*;
Ricevimento.CodM è in integrità referenziale con *Medici.CodM*;
Ricevimento.{*GiornoSet*, *OraInizio*, *OraFine*} è in integrità referenziale con *FasceOrarie.*{*GiornoSet*, *OraInizio*, *OraFine*};
Visite.Paziente è in integrità referenziale con *Pazienti.CodP*;
Visite.Medico è in integrità referenziale con *Medici.CodM*;
DettaglioVisita.Visita è in integrità referenziale con *Visite.IdV*;
DettaglioVisita.Prestazione è in integrità referenziale con *Prestazioni.CodPrest*.

Soluzione alternativa con file XML

pazienti.xml

```
?xml version="1.0"?>
<pazienti>
   <paziente codP="...">
      <cognome>...</cognome>
      <nome>...</nome>
      <codiceFisc>...</codiceFisc>
      <indirizzo>...</indirizzo>
   </paziente>
   ... altri elementi paziente
</pazienti>
```

Loescher Editore - Vietata la vendita e la diffusione

medici.xml

```
<?xml version="1.0"?>
<medici>
    <medico codM="...">
        <nome>...</nome>
        <competenze>
            <specializzazione>...</specializzazione>
            ... altri elementi specializzazione
        </competenze>
        <prestazioni>
            <prestazione codPrest="..." descrizione="..."
                 costo="..." durata="..."/>
            ... altri elementi prestazione
        </prestazioni>
        <orari>
            <giornata giorno="..." fasciaOraria="..."/>
            ... altri elementi giornata
        </orari>
    </medico>
    ... altri elementi paziente medico
</medici>
```

visite.xml

```
<?xml version="1.0"?>
<visite>
    <visita effettuata="...">
        <data>...</data>
        <ora>...</ora>
        <tempoPrevisto>...</tempoPrevisto>
        <codPaziente>...</codPaziente>
        <codMedico>...</codMedico>
        <prestazioni>
             <prestazione codPrest="..." descrizione="..."
                 costo="..." durata="..."/>
            ... altri elementi prestazione
        </prestazioni>
    </visita>
    ... altri elementi visita
</visite>
```

Nota

La soluzione fornita non è l'unica realizzabile; si può semplificare/complicare a piacere la struttura dei file, far diventare tag degli attributi, o viceversa.

Elaborare, dopo aver descritto le caratteristiche dell'applicativo che si intende utilizzare, il progetto definendone i dati necessari e dettagliando la procedura per risolvere la prima e due richieste tra quelle indicate:

1. elenco giornaliero delle visite prenotate per ogni singolo medico;
2. elenco giornaliero delle visite prenotate e non effettuate;
3. elenco settimanale contenente gli appuntamenti di ciascun medico suddivisi per giorno e per ora;
4. elenco cronologico delle visite usufruite da ciascun paziente.

Per eseguire le interrogazioni richieste, partendo dai file XML è possibile realizzare semplicemente dei fogli stile XSL che possono essere applicati usando un'applicazione server side, per esempio in JSP.

Loescher Editore - Vietata la vendita e la diffusione

Elenco giornaliero delle visite prenotate per ogni singolo medico

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:variable name="scelta">2008/11/28</xsl:variable>
<xsl:template match="visite">
    <html><head><title>Elenco giornaliero delle visite per ciascun medico</title></head>
    <body>
    <h2>Elenco giornaliero delle visite per ciascun medico</h2>
    <table border="1">
        <tr><th>Medico</th><th>Ora</th><th>Durata</th><th>Paziente</th></tr>
        <xsl:for-each select="visita">
            <xsl:sort select="codMedico"/>
            <xsl:sort select="ora"/>
            <xsl:if test="data=$scelta">
                <tr>
                    <td><xsl:value-of select="codMedico"/></td>
                    <td><xsl:value-of select="ora"/></td>
                    <td><xsl:value-of select="tempoPrevisto"/></td>
                    <td><xsl:value-of select="codPaziente"/></td>
                </tr>
            </xsl:if>
        </xsl:for-each>
    </table>
    </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

Elenco giornaliero delle visite per ciascun medico

Medico	Ora	Durata	Paziente
Medico1	09:00	1 ora e 15 min	Paziente7
Medico1	10:15	30 min	Paziente5
Medico2	08:30	30 min	Paziente2
Medico2	18:00	40 min	Paziente1

Elenco giornaliero delle visite prenotate e non effettuate

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:variable name="scelta">2008/11/28</xsl:variable>
<xsl:template match="visite">
    <html><head><title>Elenco delle visite giornaliere non effettuate</title></head>
    <body>
    <h2>Elenco delle visite giornaliere non effettuate</h2>
    <table border="1">
        <tr><th>Ora</th><th>Medico</th><th>Paziente</th><th>Durata</th></tr>
        <xsl:for-each select="visita">
            <xsl:sort select="data"/>
            <xsl:sort select="ora"/>
            <xsl:if test="data=$scelta">
                <xsl:if test="@effettuata = 'falso'">
                <tr>
                    <td><xsl:value-of select="ora"/></td>
                    <td><xsl:value-of select="codMedico"/></td>
                    <td><xsl:value-of select="codPaziente"/></td>
                    <td><xsl:value-of select="tempoPrevisto"/></td>
                </tr>
                </xsl:if>
            </xsl:if>
```

Loescher Editore - Vietata la vendita e la diffusione

```
        </xsl:for-each>
    </table>
     </body>
     </html>
</xsl:template>
</xsl:stylesheet>
```

Elenco delle visite giornaliere non effettuate

Ora	Medico	Paziente	Durata
09:00	Medico1	Paziente7	1 ora e 15 min
18:00	Medico2	Paziente1	40 min

Realizzare, infine, la pagina *web* con la quale lo studio medico pubblicizza la propria attività fornendo l'indicazione dei servizi e il quadro orario.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="medici">
    <html> <head><title>Le nostre attivit&#224;</title></head>
    <body>
    <h2>Le nostre attivit&#224;</h2>
    <table border="1">
        <tr><th>Medico</th><th>Prestazioni</th><th>Orario</th></tr>
        <xsl:for-each select="medico">
            <tr>
                <td><xsl:value-of select="nome"/></td>
                <td>
                    <table>
                        <xsl:for-each select="prestazioni/prestazione">
                            <tr><td><xsl:value-of select="@codPrest"/>,
                            <xsl:value-of select="@costo"/>,
                            <xsl:value-of select="@durata"/></td></tr>
                        </xsl:for-each>
                        </table>
                </td>
                <td>
                    <table>
                        <xsl:for-each select="orari/giornata">
                            <tr><td><xsl:value-of select="@giorno"/>,
                            <xsl:value-of select="@fasciaOraria"/></td></tr>
                        </xsl:for-each>
                    </table>
                </td>
            </tr>
        </xsl:for-each>
    </table>
    </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

Le nostre attività

Medico	Prestazioni	Orario
Nome1 Cognome1	Prestazione1, 20,00 €, 60 min Prestazione3, 15,50 €, 45 min Prestazione6, 24,00 €, 30 min	lunedì, 15:30-19:30 martedì, 8:00-13:00 e 15:30-19:30 mercoledì, 8:00-13:00 e 15:30-19:30 giovedì, 8:00-13:00 e 15:30-19:30 venerdì, 8:00-14:00
Nome2 Cognome2	Prestazione1, 20,00 €, 60 min Prestazione8, 37,00 €, 30 min Prestazione9, 7,00 €, 10 min	martedì, 8:30-13:00 e 15:30-19:30 mercoledì, 8:30-13:00 e 15:30-19:30 giovedì, 8:30-13:00 e 15:30-19:30 venerdì, 8:30-13:00 e 15:30-19:30 sabato, 8:30-12:30

Loescher Editore - Vietata la vendita e la diffusione

14.3 PORTALE PER LA VENDITA DI LIBRI

Un negozio on line chiede che sia progettato e realizzato un database per l'organizzazione e la gestione di un portale per la vendita di libri su Internet.

Il negozio richiede che:

- il Catalogo dei libri sia organizzato per:
  - Reparti (in ordine alfabetico: *architettura e urbanistica, arte, classici greci e latini, cucina e casa, diritto, economia e management, fantascienza e fantasy, filosofia, fumetti, ...*)
- ciascun Reparto sia organizzato in libri:
  - "Da non perdere" caratterizzati da uno sconto del 20% sul prezzo di copertina
  - "I più venduti" caratterizzati dal maggior numero di copie vendute negli ultimi 30 giorni
  - "Ultimi arrivi" caratterizzati dai titoli aggiunti negli ultimi 30 giorni
  - "Offerte speciali" caratterizzate da uno sconto del 25% sul prezzo di copertina
  - "Remainders" caratterizzati da uno sconto del 50% sul prezzo di copertina
- ciascun libro in negozio sia archiviato con le seguenti ulteriori informazioni:
  - titolo, autore, prezzo di copertina, sconto (eventuale), dati (numero di pagine, rilegato, ...), anno di pubblicazione, editore, collana, immagine (foto della copertina), data di archiviazione
- gli utenti abilitati all'acquisto on line siano registrati con i seguenti dati:
  - nome e cognome, indirizzo, codice di avviamento postale, città, nazione, telefono, fax, e-mail, password, numero di carta di credito, tipo di carta di credito (Visa, CartaSi, Mastercard, ...), data di scadenza della carta di credito
- gli utenti possano chiedere, in fase di acquisto, che il negozio emetta fattura
- il Carrello Acquisti sia organizzato come segue:
  - numero di copie acquistate, titolo, autore, prezzo, disponibilità (giorni, settimane, ...), togli dal carrello (casella di spunta)
- il riepilogo degli ordini sia organizzato come segue:
  - data dell'ordine, numero d'ordine
  - dettagli dell'ordine (riferimento articolo, quantità, titolo, prezzo di copertina, prezzo scontato, stato dell'ordine, evasione prevista dalla data dell'ordine).

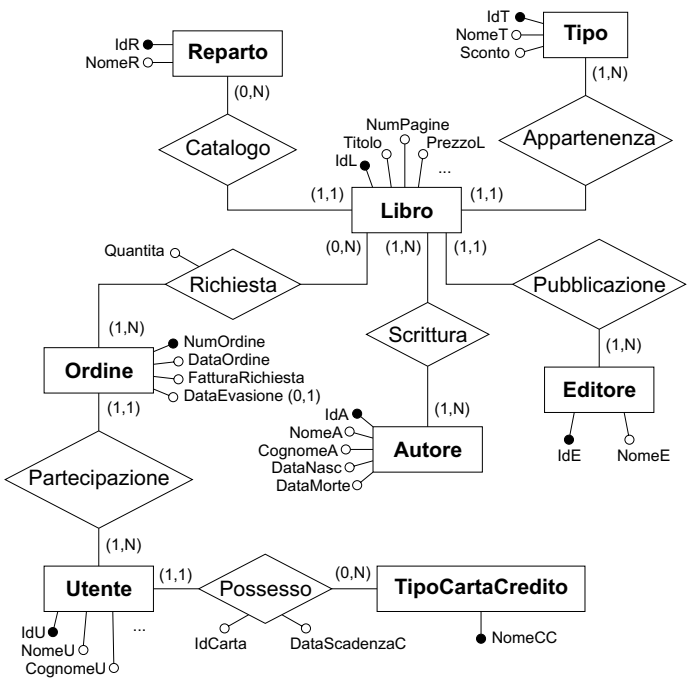
Il candidato, dopo aver fatto le eventuali ipotesi aggiuntive:

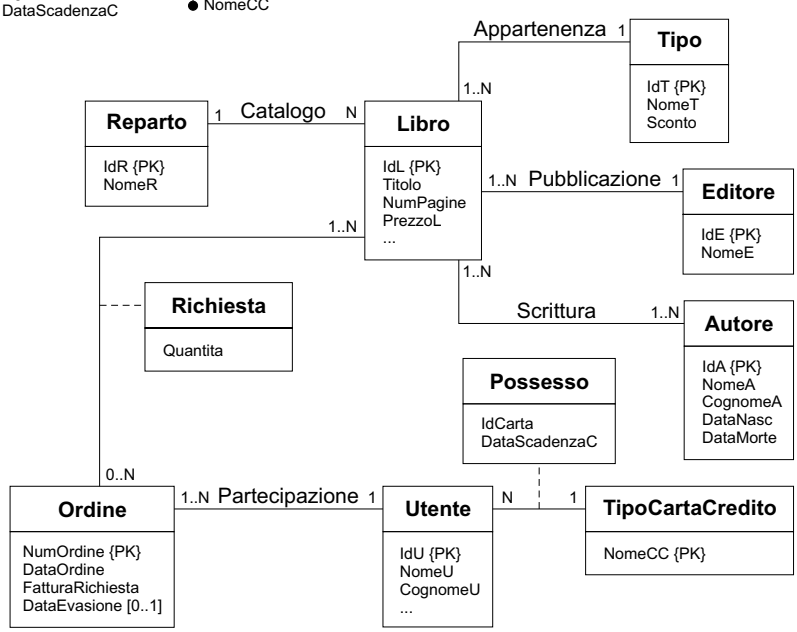
Ipotesi aggiuntive:
1) Un ordine deve essere evaso completamente o per niente.
2) Ogni utente può registrare soltanto una carta di credito.

a. Fornisca:
   1. lo schema concettuale e lo schema logico del database

Vincoli:
Dominio.1) Ordine.DataEvasione=NULL individua un ordine non evaso.
Dominio.2) Tipo.NomeT := {'Da non perdere', ...}

Loescher Editore - Vietata la vendita e la diffusione



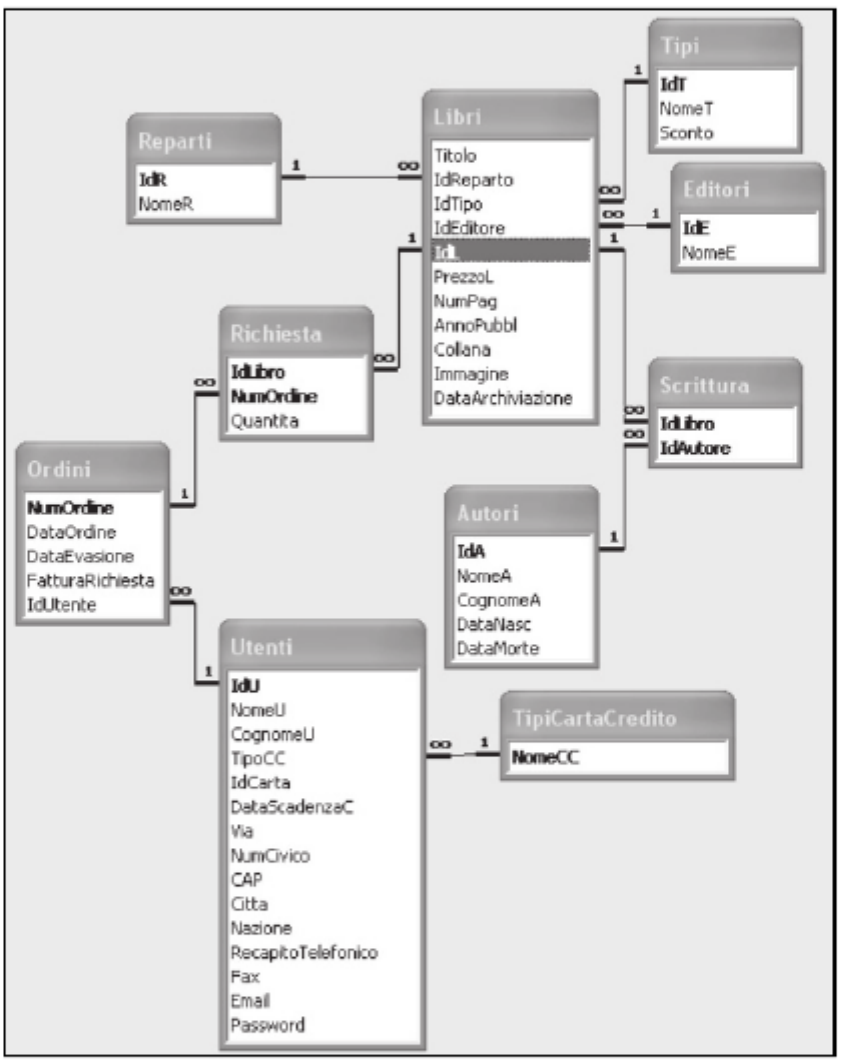


Schema relazionale

Reparti (IdR, NomeR)
Tipi (IdT, NomeT, Sconto)
Autori (IdA, NomeA, CognomeA, DataNasc, DataMorte)
Editori (IdE, NomeE)
Libri (IdL, Titolo, IdReparto, IdTipo, IdEditore, PrezzoL, NumPag, AnnoPubbl, Collana, Immagine, DataArchiviazione)
Scrittura (IdLibro, IdAutore)
TipiCartaCredito (NomeCC)
Utenti (IdU, NomeU, CognomeU, TipoCC, IdCarta, DataScadenzaC, Via, NumCivico, CAP, Citta, Nazione, RecapitoTelefonico, Fax, Email, Password)
Ordini (NumOrdine, DataOrdine, DataEvasione, FatturaRichiesta, IdUtente)
Richiesta (IdLibro, NumOrdine, Quantita)

Loescher Editore - Vietata la vendita e la diffusione

Libri.IdReparto è in integrità referenziale con *Reparti.IdR*;
Libri.IdTipo è in integrità referenziale con *Tipi.IdT*;
Libri.IdEditore è in integrità referenziale con *Editori.IdE*;
Scrittura.IdLibro è in integrità referenziale con *Libri.IdL*;
Scrittura.IdAutore è in integrità referenziale con *Autori.IdA*;
Utenti.TipoCC è in integrità referenziale con *TipiCartaCredito.NomeCC*;
Ordini.IdUtente è in integrità referenziale con *Utenti.IdU*;
Richiesta.IdLibro è in integrità referenziale con *Libri.IdL*;
Richiesta.NumOrdine è in integrità referenziale con *Ordini.NumOrdine*.



2. la definizione delle relazioni in linguaggio SQL.

Soluzione in MySQL

```
create table Reparti (
    IdR    varchar(4) primary key,
    NomeR  varchar(20) not null
)ENGINE=innodb;
```

Loescher Editore - Vietata la vendita e la diffusione

```
create table Tipi (
    IdT     varchar(4) primary key,
    NomeT   enum('Da non perdere','I più venduti','Ultimi
        arrivi','Offerte speciali','Remainders') not null,
    Sconto  smallint unsigned
)ENGINE=innodb;

create table Autori (
    IdA          varchar(4) primary key,
    NomeA        varchar(20) not null,
    CognomeA     varchar(20) not null,
    DataNasc     date not null,
    DataMorte    date not null
)ENGINE=innodb;

create table Editori (
    IdE    char(4) primary key,
    NomeE  varchar(20) not null
)ENGINE=innodb;

create table Libri (
    IdL                varchar(4) primary key,
    Titolo             varchar(30) not null,
    IdReparto          varchar(4) not null,
    IdTipo             varchar(4) not null,
    IdEditore          varchar(4) not null,
    PrezzoL            float(4,2) not null,
    NumPag             integer unsigned not null,
    AnnoPubbl          integer unsigned not null,
    Collana            varchar(30),
    Immagine           varchar(30) not null,
    DataArchiviazione  date not null,
    foreign key (IdReparto) references Reparti(IdR),
    foreign key (IdTipo) references Tipi(IdT),
    foreign key (IdEditore) references Editori(IdE)
)ENGINE=innodb;

create table Scrittura (
    IdLibro   varchar(4),
    IdAutore  varchar(4),
    primary key(IdLibro,IdAutore),
    foreign key (IdLibro) references Libri(IdL),
    foreign key (IdAutore) references Autori(IdA)
)ENGINE=innodb;

create table TipiCartaCredito (
    NomeCC  varchar(20) primary key
)ENGINE=innodb;

create table Utenti (
    IdU       varchar(10) primary key,
    NomeU     varchar(20) not null,
    CognomeU  varchar(20) not null,
    TipoCC    varchar(20) not null,
```

Loescher Editore - Vietata la vendita e la diffusione

```
    IdCarta              varchar(16) not null,
    DataScadenzaC        date not null,
    Via                  varchar(20),
    NumCivico            varchar(5),
    CAP                  varchar(5),
    Citta                varchar(20),
    Nazione              varchar(20),
    RecapitoTelefonico   varchar(14),
    Fax                  varchar(14),
    Email                varchar(30),
    Password             varchar(10) not null,
    foreign key (TipoCC) references TipiCartaCredito(NomeCC)
)ENGINE=innodb;

create table Ordini (
    NumOrdine          integer unsigned primary key,
    DataOrdine         date not null,
    DataEvasione       date,
    FatturaRichiesta   boolean not null,
    IdUtente           varchar(10) not null,
    foreign key (IdUtente) references Utenti(IdU)
)ENGINE=innodb;

create table Richiesta (
    IdLibro     varchar(4)
    NumOrdine   integer unisigned not null,
    Quantita    integer unisigned not null defult '1',
    primary key(IdLibro,NumOrdine),
    foreign key (IdLibro) references Libri(IdL),
    foreign key (NumOrdine) references Ordini(NumOrdine)
)ENGINE=innodb;
```

b. Implementi le seguenti query:

1. *Ricerca per Reparto*: scelto un Reparto, il numero di libri “Da non perdere” ed il numero di libri “I più venduti” con i relativi dettagli

Numero di libri “Da non perdere” e numero di libri “I più venduti” del reparto scelto

```
SELECT R.NomeR, T.NomeT, count(T.NomeT) AS NumeroDiLibri
FROM Reparti AS R, Libri AS L, Tipi AS T
WHERE (R.IdR=L.IdReparto) AND (L.IdTipo=T.IdT) AND
      (R.NomeR=[Inserisci il nome del reparto:]) AND
      ((T.NomeT='Da non perdere') OR (T.NomeT='I più venduti'))
GROUP BY R.NomeR, T.NomeT;
```

Dettaglio dei libri “Da non perdere” e “I più venduti” del reparto scelto

```
SELECT R.NomeR, T.NomeT, L.Titolo, L.PrezzoL AS Prezzo, L.NumPag
FROM Reparto AS R, Libro AS L, Tipo AS T
WHERE (R.IdR=L.IdReparto) AND (L.IdTipo=T.IdT) AND
       (R.NomeR=[Inserisci il nome del reparto:]) AND
       ((T.NomeT='Da non perdere') Or (T.NomeT='I più venduti'))
ORDER BY T.NomeT, L.Titolo;
```

Loescher Editore - Vietata la vendita e la diffusione

2. *Ordini*: gli ordini in corso, con dettagli, di un dato utente.

```
SELECT U.NomeU, U.CognomeU, U.IdU, O.NumOrdine, O.DataOrdine,
L.Titolo AS TitoloLibro, R.Quantita
FROM Utenti AS U, Ordini AS O, Richiesta AS R, Libri AS L
WHERE (O.IdUtente=U.IdU) AND (O.NumOrdine=R.NumOrdine) AND
  (R.IdLibro=L.IdL) AND (O.IdUtente=[Inserisci l'ID di un utente:])
  AND (O.DataEvasione IS NULL)
ORDER BY O.NumOrdine;
```

c. Scriva in un linguaggio lato server, il codice di almeno una delle seguenti pagine del portale:
   1. con accesso libero, la pagina utile a visualizzare i Reparti e, per ciascun reparto, la pagina che implementa la query n. 1;
   2. con accesso riservato agli utenti registrati, la composizione degli ordini in corso.

Soluzione del punto 2 in JSP

index.jsp

```
<%@ page language="java" import="java.sql.*" %>
<html>
<head><title>Indentificazione</title></head>
<body>
<h2>Login</h2>
<h3>Inserisci IdUtente e Password per visualizzare gli ordini in corso</h3>
<form action="visualizza.jsp" method="post">
<table>
    <tr>
        <td style="text-align:right">IdUtente:</td>
        <td><input type="text" name="idUtente"/></td>
    </tr>
    <tr>
        <td style="text-align:right">Password:</td>
        <td><input type="password" name="password"/></td>
    </tr>
</table>
<br/><input type="submit" value="Entra">
</form>
</body>
</html>
```

visualizza.jsp

```
<%@ page language="java" import="java.sql.*" %>
<html>
<head><title>Ordini in corso</title></head>
<body>
 <%
    String id=request.getParameter("idUtente");
    String pswInserita=request.getParameter("password");
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection con = DriverManager.getConnection("jdbc:odbc:Libri");
    Statement st = con.createStatement();
```

Loescher Editore - Vietata la vendita e la diffusione

```
        String query1="SELECT Password, NomeU, CognomeU FROM Utenti WHERE IdU='"+id+"'";
        ResultSet rs = st.executeQuery(query1);
        if(rs.next()){ //se l'id è presente nella tabella si può controllare la password
            String pswGiusta=rs.getString("Password");
            String nome=rs.getString("NomeU");
            String cognome=rs.getString("CognomeU");
            if(pswGiusta.equals(pswInserita)){
            //idUtente e password corretti, si procede alla visualizzazione degli ordini
                out.print("<h2>Ordini in corso relativi a "+nome+" "+cognome+"</h2>");
                rs.close(); %>
                <table border="1">
                    <tr>
                        <td><b>Num. ordine</b></td>
                        <td><b>Data dell'ordine</b></td>
                        <td><b>Titolo del libro</b></td>
                        <td><b>Quantita'</b></td>
                        <td><b>Prezzo Unitario</b></td>
                    </tr>
                <%
                String ordini="SELECT O.NumOrdine, O.DataOrdine, L.Titolo AS TitoloLibro, R.Quantita, L.PrezzoL";
                ordini+=" FROM Ordini AS O, Utenti AS U, Richiesta AS R, Libri AS L";
                ordini+=" WHERE (O.IdUtente='"+id+"') AND (O.NumOrdine=R.NumOrdine)";
                ordini+=" AND (O.IdUtente=U.IdU AND R.IdLibro=L.IdL) AND (O.DataEvasione IS NULL)";
                ordini+=" ORDER BY O.NumOrdine";
                ResultSet rsOrd=st.executeQuery(ordini);
                while (rsOrd.next()) {
                    out.println("<tr>");
                    out.println("<td>" + rsOrd.getString("NumOrdine") + "</td>");
                    out.println("<td>" + rsOrd.getString("DataOrdine") + "</td>");
                    out.println("<td>" + rsOrd.getString("TitoloLibro") + "</td>");
                    out.println("<td>" + rsOrd.getString("Quantita") + "</td>");
                    out.println("<td>" + rsOrd.getDouble("PrezzoL") + "</td>");
                    out.println("</tr>");
                }//while
                out.print("<table><br/>");
                rsOrd.close();
            }//if
            else{
                out.print("<h2>Attenzione, password errata!</h2>");
            }//else
        }//if
        else
            out.print("<h2>Attenzione, nome utente errato!</h2>");
        st.close();
        con.close();
%>
<form action="index.jsp" method="post">
    <input type="submit" value="Indietro">
</form>
</body>
</html>
```

Ordini in corso relativi a NomeU1 CognomeU1

Num. ordine	Data dell'ordine	Titolo del libro	Quantita'	Prezzo Unitario
2	2008-11-18 00:00:00	TitoloL2	3	17.0

Indietro

Loescher Editore - Vietata la vendita e la diffusione

14.4 COMMUNITY DI APPRENDIMENTO SUL WEB

Una rete di scuole chiede che sia progettato e realizzato un database per l'organizzazione e la gestione del portale di una community di apprendimento sul Web.

L'organizzazione della community prevede che:

- l'accesso sia consentito ai soli utenti registrati;
- gli utenti siano distribuiti in tre gruppi: amministratore, docente, studente tali che:
  - un amministratore abbia accesso a tutte le aree protette del portale;
  - un docente abbia accesso a tutte le aree protette tranne che all'area di amministrazione;
  - uno studente abbia accesso alla propria area protetta e non abbia accesso né all'area di amministrazione né all'area riservata al gruppo docente;
- la registrazione degli utenti consenta:
  - alla rete di scuole di acquisire informazioni, sotto forma di dati non sensibili, relative agli utenti quali, ad esempio, nome e cognome, scuola o istituzione formativa di appartenenza, collocazione geografica, e-mail, ecc….
  - agli utenti di scegliere un nome utente, una password e il gruppo di appartenenza tranne quello degli amministratori;
- agli utenti registrati, ciascuno per il proprio gruppo di appartenenza, sia consentito di effettuare l'upload di documenti multimediali archiviando:
  - il titolo
  - il tipo di documento (testo, audio, ecc.)
  - una descrizione sintetica
  - la data di upload
  - i dati personali che lo riguardano;
- il gruppo degli studenti possa usufruire di moduli formativi ad esso rivolti;
- il gruppo dei docenti possa usufruire sia di propri moduli formativi che di quelli rivolti agli studenti;
- ciascun modulo formativo sia individuato da un titolo, da una breve descrizione e dal tipo di utente cui è rivolto.

Il candidato, fatte eventuali ipotesi aggiuntive:

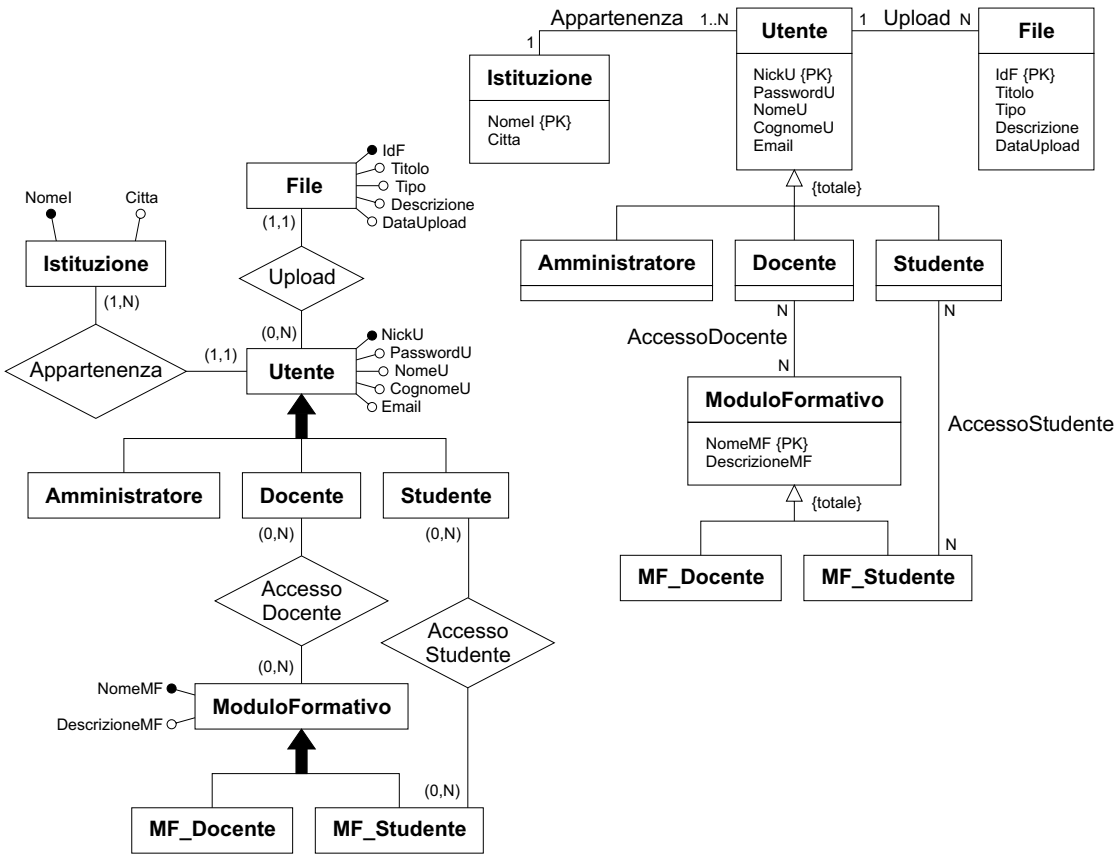
La gestione dei diversi tipi di utenti potrebbe essere realizzata configurando il server Web in modo da richiedere l'autenticazione e raccogliendo le pagine riservate in aree opportune con accesso limitato.

- fornisca:
  1. Lo schema concettuale e lo schema logico del database;

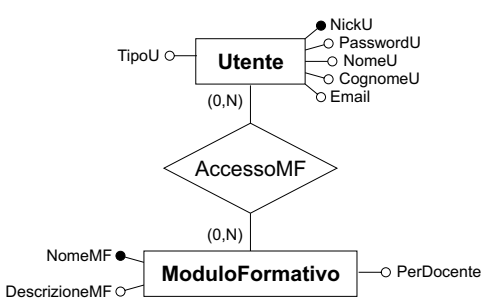
Vincoli:
1) L'utente può essere di 3 tipi: amministratore, docente, studente.
2) Gli utenti che risultano essere studenti possono accedere solo ai moduli formativi rivolti agli studenti (con il campo *PerDocente* impostato a *false*).

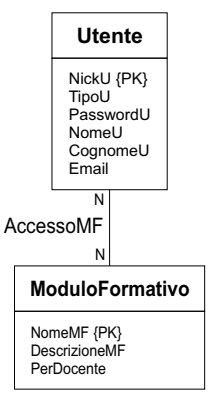
Dominio.1) File.Tipo := {'Testo', 'Audio', ...}
Dominio.2) Utenti.TipoU := {'amministratore', 'docente', 'studente'}

Loescher Editore - Vietata la vendita e la diffusione



La generalizzazione si può ristrutturare in questo modo:





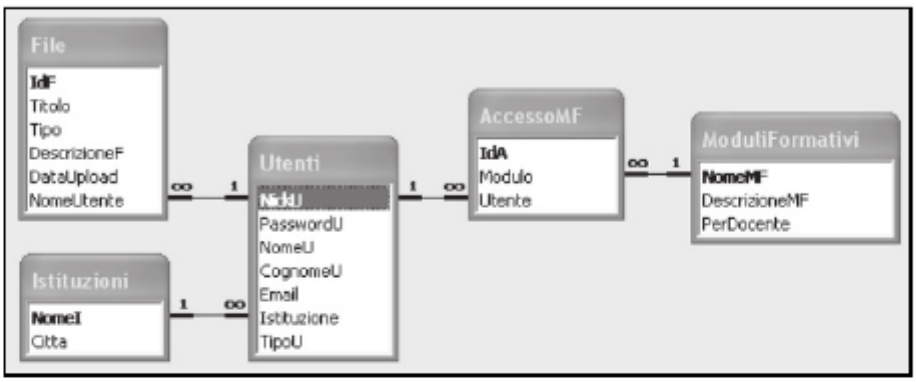
Schema relazionale

Istituzioni (NomeI, Citta)
Utenti (NickU, PasswordU, NomeU, CognomeU, Email, Istituzione, TipoU)
File (IdF, Titolo, Tipo, Descrizione, DataUpload, NomeUtente)
ModuliFormativi (NomeMF, DescrizioneMF, PerDocente)
AccessoMF (Modulo, Utente)

Si può sostituire la relazione *AccessoMF* con:
AccessoMF (IdA, Modulo, Utente)
in modo da poter realizzare un log degli accessi, registrando anche più accessi di un utente allo stesso modulo.

Loescher Editore - Vietata la vendita e la diffusione

Utenti.Istituzione è in integrità referenziale con *Istituzioni.NomeI*;
File.NomeUtente è in integrità referenziale con *Utenti.NickU*;
AccessoMF.Modulo è in integrità referenziale con *ModuliFormativi.NomeMF*;
AccessoMF.Utente è in integrità referenziale con *Utenti.NickU*.



2. La definizione delle relazioni in linguaggio SQL.

Soluzione in MySQL

```
create table Istituzioni (
    NomeI  varchar(20) primary key,
    Citta  varchar(20) not null
)ENGINE=innodb;

create table Utenti (
    NickU        varchar(20) primary key,
    PasswordU    varchar(10) not null,
    NomeU        varchar(20) not null,
    CognomeU     varchar(20) not null,
    Email        varchar(30),
    Istituzione  varchar(20) not null,
    TipoU        enum('amministratore','docente','studente') not null,
    foreign key (Istituzione) references Istituzioni(NomeI)
)ENGINE=innodb;

create table File (
    IdF          integer unsigned primary key,
    Titolo       varchar(20) not null,
    Tipo         enum('Testo','Audio','Immagine','Filmato') not null,
    Descrizione  varchar(100),
    DataUpload   date not null,
    NomeUtente   varchar(20) not null,
    foreign key (NomeUtente) references Utenti(NickU)
)ENGINE=innodb;

create table ModuliFormativi (
    NomeMF         varchar(20) primary key,
    DescrizioneMF  varchar(150) not null,
    PerDocente     boolean not null
)ENGINE=innodb;

create table AccessoMF (
    IdA     integer unsigned primary key,
```

Loescher Editore - Vietata la vendita e la diffusione

```
    Modulo  varchar(20) not null,
    Utente  varchar(20) not null,
    foreign key (Modulo) references ModuliFormativi(NomeMF),
    foreign key (Utente) references Utenti(NickU)
)ENGINE=innodb;
```

- implementi almeno una delle seguenti query:

  n. 1: I docenti che hanno un account presso la community con la rispettiva collocazione geografica ed i moduli formativi scelti.

```
SELECT DISTINCT U.NomeU, U.CognomeU, U.NickU, I.NomeI, I.Citta,
MF.NomeMF, MF.DescrizioneMF
FROM ModuliFormativi AS MF, AccessoMF AS AMF, Utenti AS U,
Istituzioni AS I
WHERE (MF.NomeMF=AMF.Modulo) AND (AMF.Utente=U.NickU) AND
      (U.Istituzione=I.NomeI) AND (U.TipoU='docente')
ORDER BY U.NickU, MF.NomeMF;
```

n. 2: I dati relativi agli studenti e ai documenti che essi hanno inviato in remoto sul portale della community mediante upload.

```
SELECT U.NomeU, U.CognomeU, U.NickU, U.Email, U.Istituzione,
F.Titolo, F.Tipo, F.DescrizioneF, F.DataUpload
FROM Utenti AS U INNER JOIN File AS F ON U.NickU=F.NomeUtente
WHERE U.TipoU='studente'
ORDER BY U.NickU, F.DataUpload;
```

- scriva, in un linguaggio lato server, il codice di almeno una delle seguenti pagine del portale:
  - con accesso riservato all'amministratore, il report che trae i dati dalla query n. 1
  - con accesso riservato agli studenti, il report che trae i dati dalla query n. 2
  - registrazione di un nuovo utente con eventuale invio automatico dei dati registrati mediante e-mail diretta all'utente appena registrato e ad un amministratore.

Per mandare una email da una applicazione server side bisogna utilizzare funzioni specifiche in base alla tecnologia utilizzata.

Per esempio in PHP la funzione per spedire messaggi di posta elettronica nella forma più semplice è:

mail (destinatario, mittente, testo);

Istruzioni tipo per spedire un messaggio sono:

```
<?php
    $destinatario = ...
    $oggetto = ...
    $testo = ...
    mail ($destinatario, $oggetto, $testo)
?>
```

Per inviare lo stesso messaggio a più destinatari si possono inserire tutti gli indirizzi nella variabile *$destinatario*, separati da virgole.

La funzione restituisce *true* se il messaggio è stato inviato correttamente.

Loescher Editore - Vietata la vendita e la diffusione

Appendice

A Database e diritto

Loescher Editore - Vietata la vendita e la diffusione

A DATABASE E DIRITTO

TUTELA DEI DATI PERSONALI

Il trattamento dei dati personali è disciplinato dal **Decreto legislativo 196 del 30 giugno 2003** "Codice in materia di protezione dei dati personali".

Nota

In precedenza il trattamento dei dati personali era disciplinato dalla legge 675 del 31 dicembre 1996 "Tutela delle persone e di altri soggetti rispetto al trattamento dei dati personali" nota anche come legge sulla privacy.

Le parti più significative del decreto 196 sono:

Art. 1 - Diritto alla protezione dei dati personali

1. Chiunque ha diritto alla protezione dei dati personali che lo riguardano.

Art. 2 - Finalità

1. Il presente testo unico, di seguito denominato "codice", garantisce che il trattamento dei dati personali si svolga nel rispetto dei diritti e delle libertà fondamentali, nonché della dignità dell'interessato, con particolare riferimento alla riservatezza, all'identità personale e al diritto alla protezione dei dati personali.

Art. 3 - Principio di necessità nel trattamento dei dati

1. I sistemi informativi e i programmi informatici sono configurati riducendo al minimo l'utilizzazione di dati personali e di dati identificativi, in modo da escluderne il trattamento quando le finalità perseguite nei singoli casi possono essere realizzate mediante, rispettivamente, dati anonimi od opportune modalità che permettano di identificare l'interessato solo in caso di necessità.

Art. 4 - Definizioni

1. Ai fini del presente codice si intende per:

a) "**trattamento**", qualunque operazione o complesso di operazioni, effettuati anche senza l'ausilio di strumenti elettronici, concernenti la raccolta, la registrazione, l'organizzazione, la conservazione, la consultazione, l'elaborazione, la modificazione, la selezione, l'estrazione, il raffronto, l'utilizzo, l'interconnessione, il blocco, la comunicazione, la diffusione, la cancellazione e la distruzione di dati, anche se non registrati in una banca di dati;

b) "**dato personale**", qualunque informazione relativa a persona fisica, persona giuridica, ente od associazione, identificati o identificabili, anche indirettamente, mediante riferimento a qualsiasi altra informazione, ivi compreso un numero di identificazione personale;

c) "**dati identificativi**", i dati personali che permettono l'identificazione diretta dell'interessato;

d) "**dati sensibili**", i dati personali idonei a rivelare l'origine razziale ed etnica, le convinzioni religiose, filosofiche o di altro genere, le opinioni politiche, l'adesione a partiti, sindacati, associazioni od organizzazioni a carattere religioso, filosofico, politico o sindacale, nonché i dati personali idonei a rivelare lo stato di salute e la vita sessuale;

e) "**dati giudiziari**", i dati personali idonei a rivelare provvedimenti di cui all'articolo 3, comma 1, lettere da a) a o) e da r) a u), del D.P.R. 14 novembre 2002, n. 313, in materia

Loescher Editore - Vietata la vendita e la diffusione

di casellario giudiziale, di anagrafe delle sanzioni amministrative dipendenti da reato e dei relativi carichi pendenti, o la qualità di imputato o di indagato ai sensi degli articoli 60 e 61 del codice di procedura penale;
f) "**titolare**", la persona fisica, la persona giuridica, la pubblica amministrazione e qualsiasi altro ente, associazione od organismo cui competono, anche unitamente ad altro titolare, le decisioni in ordine alle finalità, alle modalità del trattamento di dati personali e agli strumenti utilizzati, ivi compreso il profilo della sicurezza;
g) "**responsabile**", la persona fisica, la persona giuridica, la pubblica amministrazione e qualsiasi altro ente, associazione od organismo preposti dal titolare al trattamento di dati personali;
h) "**incaricati**", le persone fisiche autorizzate a compiere operazioni di trattamento dal titolare o dal responsabile;
i) "**interessato**", la persona fisica, la persona giuridica, l'ente o l'associazione cui si riferiscono i dati personali;
l) "**comunicazione**", il dare conoscenza dei dati personali a uno o più soggetti determinati diversi dall'interessato, dal rappresentante del titolare nel territorio dello Stato, dal responsabile e dagli incaricati, in qualunque forma, anche mediante la loro messa a disposizione o consultazione;
m) "**diffusione**", il dare conoscenza dei dati personali a soggetti indeterminati, in qualunque forma, anche mediante la loro messa a disposizione o consultazione;
n) "**dato anonimo**", il dato che in origine, o a seguito di trattamento, non può essere associato ad un interessato identificato o identificabile;
o) "**blocco**", la conservazione di dati personali con sospensione temporanea di ogni altra operazione del trattamento;
p) "**banca di dati**", qualsiasi complesso organizzato di dati personali, ripartito in una o più unità dislocate in uno o più siti;
q) "**Garante**", l'autorità di cui all'articolo 153, istituita dalla legge 31 dicembre 1996, n. 675.

3. Ai fini del presente codice si intende, altresì, per:
a) "**misure minime**", il complesso delle misure tecniche, informatiche, organizzative, logistiche e procedurali di sicurezza che configurano il livello minimo di protezione richiesto in relazione ai rischi previsti nell'articolo 31;
b) "**strumenti elettronici**", gli elaboratori, i programmi per elaboratori e qualunque dispositivo elettronico o comunque automatizzato con cui si effettua il trattamento;
c) "**autenticazione informatica**", l'insieme degli strumenti elettronici e delle procedure per la verifica anche indiretta dell'identità;
d) "**credenziali di autenticazione**", i dati ed i dispositivi, in possesso di una persona, da questa conosciuti o ad essa univocamente correlati, utilizzati per l'autenticazione informatica;
e) "**parola chiave**", componente di una credenziale di autenticazione associata ad una persona ed a questa nota, costituita da una sequenza di caratteri o altri dati in forma elettronica;
f) "**profilo di autorizzazione**", l'insieme delle informazioni, univocamente associate ad una persona, che consente di individuare a quali dati essa può accedere, nonché i trattamenti ad essa consentiti;
g) "**sistema di autorizzazione**", l'insieme degli strumenti e delle procedure che abilitano l'accesso ai dati e alle modalità di trattamento degli stessi, in funzione del profilo di autorizzazione del richiedente.

Art. 7 - Diritto di accesso ai dati personali ed altri diritti
1. L'interessato ha diritto di ottenere la conferma dell'esistenza o meno di dati personali che lo riguardano, anche se non ancora registrati, e la loro comunicazione in forma intelligibile.
2. L'interessato ha diritto di ottenere l'indicazione:
a) dell'origine dei dati personali;

Loescher Editore - Vietata la vendita e la diffusione

b) delle finalità e modalità del trattamento;
c) della logica applicata in caso di trattamento effettuato con l'ausilio di strumenti elettronici;
d) degli estremi identificativi del titolare, dei responsabili e del rappresentante designato ai sensi dell'articolo 5, comma 2;
e) dei soggetti o delle categorie di soggetti ai quali i dati personali possono essere comunicati o che possono venirne a conoscenza in qualità di rappresentante designato nel territorio dello Stato, di responsabili o incaricati.
3. L'interessato ha diritto di ottenere:
a) l'aggiornamento, la rettificazione ovvero, quando vi ha interesse, l'integrazione dei dati;
b) la cancellazione, la trasformazione in forma anonima o il blocco dei dati trattati in violazione di legge, compresi quelli di cui non è necessaria la conservazione in relazione agli scopi per i quali i dati sono stati raccolti o successivamente trattati;
c) l'attestazione che le operazioni di cui alle lettere a) e b) sono state portate a conoscenza, anche per quanto riguarda il loro contenuto, di coloro ai quali i dati sono stati comunicati o diffusi, eccettuato il caso in cui tale adempimento si rivela impossibile o comporta un impiego di mezzi manifestamente sproporzionato rispetto al diritto tutelato.
4. L'interessato ha diritto di opporsi, in tutto o in parte:
a) per motivi legittimi al trattamento dei dati personali che lo riguardano, ancorché pertinenti allo scopo della raccolta;
b) al trattamento di dati personali che lo riguardano a fini di invio di materiale pubblicitario o di vendita diretta o per il compimento di ricerche di mercato o di comunicazione commerciale.

Art. 11 - Modalità del trattamento e requisiti dei dati
1. I dati personali oggetto di trattamento sono:
a) trattati in modo lecito e secondo correttezza;
b) raccolti e registrati per scopi determinati, espliciti e legittimi, ed utilizzati in altre operazioni del trattamento in termini compatibili con tali scopi;
c) esatti e, se necessario, aggiornati;
d) pertinenti, completi e non eccedenti rispetto alle finalità per le quali sono raccolti o successivamente trattati;
e) conservati in una forma che consenta l'identificazione dell'interessato per un periodo di tempo non superiore a quello necessario agli scopi per i quali essi sono stati raccolti o successivamente trattati.
2. I dati personali trattati in violazione della disciplina rilevante in materia di trattamento dei dati personali non possono essere utilizzati.

Art. 13 - Informativa
1. L'interessato o la persona presso la quale sono raccolti i dati personali sono previamente informati oralmente o per iscritto circa:
a) le finalità e le modalità del trattamento cui sono destinati i dati;
b) la natura obbligatoria o facoltativa del conferimento dei dati;
c) le conseguenze di un eventuale rifiuto di rispondere;
d) i soggetti o le categorie di soggetti ai quali i dati personali possono essere comunicati o che possono venirne a conoscenza in qualità di responsabili o incaricati, e l'ambito di diffusione dei dati medesimi;
e) i diritti di cui all'articolo 7;
f) gli estremi identificativi del titolare e, se designati, del rappresentante nel territorio dello Stato ai sensi dell'articolo 5 e del responsabile. Quando il titolare ha designato più responsabili è indicato almeno uno di essi, indicando il sito della rete di comunicazione o le modalità attraverso le quali è conoscibile in modo agevole l'elenco aggiornato dei responsabili. Quando è stato designato un responsabile per il riscontro all'interessato in caso di esercizio dei diritti di cui all'articolo 7, è indicato tale responsabile.

Loescher Editore - Vietata la vendita e la diffusione

Art. 15 - Danni cagionati per effetto del trattamento
1. Chiunque cagiona danno ad altri per effetto del trattamento di dati personali è tenuto al risarcimento ai sensi dell'articolo 2050 del codice civile.

Art. 2050 del Codice Civile
Chiunque cagiona danno ad altri nello svolgimento di una attività pericolosa, per sua natura o per la natura dei mezzi adoperati, è tenuto al risarcimento se non prova di aver adottato tutte le misure idonee ad evitare il danno.

Nota
Per la responsabilità penale basta l'adozione delle misure minime ma per la responsabilità civile è necessaria l'adozione di misure idonee.

Art. 23 - Consenso
1. Il trattamento di dati personali da parte di privati o di enti pubblici economici è ammesso solo con il consenso espresso dell'interessato.
2. Il consenso può riguardare l'intero trattamento ovvero una o più operazioni dello stesso.
3. Il consenso è validamente prestato solo se è espresso liberamente e specificamente in riferimento ad un trattamento chiaramente individuato, se è documentato per iscritto, e se sono state rese all'interessato le informazioni di cui all'articolo 13.
4. Il consenso è manifestato in forma scritta quando il trattamento riguarda dati sensibili.

Art. 24 - Casi nei quali può essere effettuato il trattamento senza consenso
1. Il consenso non è richiesto, oltre che nei casi previsti nella Parte II, quando il trattamento:
a) è necessario per adempiere ad un obbligo previsto dalla legge, da un regolamento o dalla normativa comunitaria;
b) è necessario per eseguire obblighi derivanti da un contratto del quale è parte l'interessato o per adempiere, prima della conclusione del contratto, a specifiche richieste dell'interessato;
c) riguarda dati provenienti da pubblici registri, elenchi, atti o documenti conoscibili da chiunque, fermi restando i limiti e le modalità che le leggi, i regolamenti o la normativa comunitaria stabiliscono per la conoscibilità e pubblicità dei dati;
d) riguarda dati relativi allo svolgimento di attività economiche, trattati nel rispetto della vigente normativa in materia di segreto aziendale e industriale;
e) è necessario per la salvaguardia della vita o dell'incolumità fisica di un terzo. Se la medesima finalità riguarda l'interessato e quest'ultimo non può prestare il proprio consenso per impossibilità fisica, per incapacità di agire o per incapacità di intendere o di volere, il consenso è manifestato da chi esercita legalmente la potestà, ovvero da un prossimo congiunto, da un familiare, da un convivente o, in loro assenza, dal responsabile della struttura presso cui dimora l'interessato. Si applica la disposizione di cui all'articolo 82, comma 2;
f) con esclusione della diffusione, è necessario ai fini dello svolgimento delle investigazioni difensive di cui alla legge 7 dicembre 2000, n. 397, o, comunque, per far valere o difendere un diritto in sede giudiziaria, sempre che i dati siano trattati esclusivamente per tali finalità e per il periodo strettamente necessario al loro perseguimento, nel rispetto della vigente normativa in materia di segreto aziendale e industriale;
g) con esclusione della diffusione, è necessario, nei casi individuati dal Garante sulla base dei principi sanciti dalla legge, per perseguire un legittimo interesse del titolare o di un terzo destinatario dei dati, anche in riferimento all'attività di gruppi bancari e di società controllate o collegate, qualora non prevalgano i diritti e le libertà fondamentali, la dignità o un legittimo interesse dell'interessato;
h) con esclusione della comunicazione all'esterno e della diffusione, è effettuato da associazioni, enti od organismi senza scopo di lucro, anche non riconosciuti, in riferimento a soggetti che hanno con essi contatti regolari o ad aderenti, per il perseguimento di scopi determinati e legittimi individuati dall'atto costitutivo, dallo statuto o dal contratto

Loescher Editore - Vietata la vendita e la diffusione

collettivo, e con modalità di utilizzo previste espressamente con determinazione resa nota agli interessati all'atto dell'informativa ai sensi dell'articolo 13;
i) è necessario, in conformità ai rispettivi codici di deontologia di cui all'allegato A), per esclusivi scopi scientifici o statistici, ovvero per esclusivi scopi storici presso archivi privati dichiarati di notevole interesse storico ai sensi dell'articolo 6, comma 2, del decreto legislativo 29 ottobre 1999, n. 490, di approvazione del testo unico in materia di beni culturali e ambientali o, secondo quanto previsto dai medesimi codici, presso altri archivi privati.

Art. 26 - Garanzie per i dati sensibili
1. I dati sensibili possono essere oggetto di trattamento solo con il consenso scritto dell'interessato e previa autorizzazione del Garante, nell'osservanza dei presupposti e dei limiti stabiliti dal presente codice, nonché dalla legge e dai regolamenti.

Art. 27 - Garanzie per i dati giudiziari
1. Il trattamento di dati giudiziari da parte di privati o di enti pubblici economici è consentito soltanto se autorizzato da espressa disposizione di legge o provvedimento del Garante che specifichino le rilevanti finalità di interesse pubblico del trattamento, i tipi di dati trattati e di operazioni eseguibili.

Art. 31 - Obblighi di sicurezza
1. I dati personali oggetto di trattamento sono custoditi e controllati, anche in relazione alle conoscenze acquisite in base al progresso tecnico, alla natura dei dati e alle specifiche caratteristiche del trattamento, in modo da ridurre al minimo, mediante l'adozione di idonee e preventive misure di sicurezza, i rischi di distruzione o perdita, anche accidentale, dei dati stessi, di accesso non autorizzato o di trattamento non consentito o non conforme alle finalità della raccolta.

Art. 33 - Misure minime
1. Nel quadro dei più generali obblighi di sicurezza di cui all'articolo 31, o previsti da speciali disposizioni, i titolari del trattamento sono comunque tenuti ad adottare le misure minime individuate nel presente capo o ai sensi dell'articolo 58, comma 3, volte ad assicurare un livello minimo di protezione dei dati personali.

Art. 34 - Trattamenti con strumenti elettronici
1. Il trattamento di dati personali effettuato con strumenti elettronici è consentito solo se sono adottate, nei modi previsti dal disciplinare tecnico contenuto nell'allegato B), le seguenti misure minime:
a) autenticazione informatica;
b) adozione di procedure di gestione delle credenziali di autenticazione;
c) utilizzazione di un sistema di autorizzazione;
d) aggiornamento periodico dell'individuazione dell'ambito del trattamento consentito ai singoli incaricati e addetti alla gestione o alla manutenzione degli strumenti elettronici;
e) protezione degli strumenti elettronici e dei dati rispetto a trattamenti illeciti di dati, ad accessi non consentiti e a determinati programmi informatici;
f) adozione di procedure per la custodia di copie di sicurezza, il ripristino della disponibilità dei dati e dei sistemi;
g) tenuta di un aggiornato documento programmatico sulla sicurezza;
h) adozione di tecniche di cifratura o di codici identificativi per determinati trattamenti di dati idonei a rivelare lo stato di salute o la vita sessuale effettuati da organismi sanitari.

Art. 36 - Adeguamento
1. Il disciplinare tecnico di cui all'allegato B), relativo alle misure minime di cui al presente capo, è aggiornato periodicamente con decreto del Ministro della giustizia di concerto con il Ministro per le innovazioni e le tecnologie, in relazione all'evoluzione tecnica e all'esperienza maturata nel settore.

Loescher Editore - Vietata la vendita e la diffusione

INDICE ANALITICO

! 162, 245, 286
!< 47
!= 47, 286
!> 47
212, 286, 324
$ 286
$_COOKIE 290, 292
$_GET 290
$_POST 290, 291
$_SERVER 290, 291
$_SERVER['PHP_SELF'] 291
$_SESSION 290, 292
$GLOBALS 291
$HTTP_COOKIE_VARS 290, 292
$HTTP_GET_VARS 290
$HTTP_POST_VARS 290
$HTTP_SERVER_VARS 290
$HTTP_SESSION_VARS 290
$PHP_SELF 291
% 162, 279, 286
%= 286
%> 286, 310
& 213, 278, 343
&#n; 341
&& 162, 286
& 343
' 343
> 343
< 343
" 343
(56
) 56
* 39, 46, 50, 59, 87, 162, 211, 212, 286, 352
*/ 171, 286
*= 286
+ 39, 162, 278, 286
++ 286
+= 286
- 39, 162, 286
-- 171, 286
--> 342
-= 286
-> 158
. 245, 287, 352
.. 352
.= 287
.accdb 193
.accde 249
.aspx 314
.gif 297
.html 297
.jpeg 297
.jsp 297
.laccdb 250
.mdb 193
.mde 249
.NET 314
/ 39, 162, 286, 352
/* 171, 286
// 286, 352
/= 286
/etc/httpd/conf 283
: 343
:= 170
; 158
< 39, 47, 162, 212, 286, 343
<!-- 342
<% 286, 310
<%# 324
<%= 310
<= 39, 47, 162, 212, 286
<> 39, 47, 162, 212
<? 286, 342
<?php 285, 286
= 39, 47, 162, 278, 286
== 286
=== 286
> 39, 47, 157, 162, 212, 286, 343
>= 39, 47, 162, 212, 286
? 212
?: 287
?> 285, 286, 342
@ 170, 316, 352
@* 352
[38, 213, 353
\G 169
] 38, 213, 353
_ 38, 298
_be 250
_jspService 299
{ 38, 350
| 38, 352
|| 162, 286
} 38, 350
3NF 130, 135

A

A scelta obbligatoria 223
ABS 162
absolute 271
Access 193
AccessDataSource 328
accesso ai database 11, 257
accesso calcolato 146
accesso concorrente 66
accorpamento di entità 115
ACID 67
action 278
ActiveControl 245
ActiveDataSheet 245
ActiveForm 245
ActiveReport 245
actuate 344
ADD COLUMN 43
ADD CONSTRAINT 43
AddNew 313
adLockOptimistic 314
ADO 12, 257, 258, 311
ADO.NET 12, 257, 259, 322
AdRotator 316
affidabilità 3
AFTER 64
afterLast 271
alberi di definizione 7
albero B 147
albero B+ 148
algebra relazionale 6, 22
alias 53
ALL 46, 57, 58, 66
Allegato 199
AllForms 244
AllowPaging 329
AllQueries 244
AllReports 244
AllTables 244
ALTER DOMAIN 39
ALTER TABLE 43, 167
AlternatingItemTemplate 326
amministratore del database 12
AND 48, 162
and 286
AnnullaEvento 238, 241
anomalie 7, 21, 127
ANY 57, 58
Apache 282, 283
API JDBC 306
App_Data 316
ApplicaFiltro 238
Application 244, 310
application 299, 301
applicazioni Access 247
applicazioni server side 12
applicazioni XML 340
apply-templates 348

Loescher Editore - Vietata la vendita e la diffusione

ApriMaschera 238
ApriQuery 238
ApriReport 238
ApriTabella 238
architettura 3-tier 12, 281
architettura di MySQL 155
Array 286
array 288
array associativi 289
array superglobali 290
array_merge 289
array_push 289
array_shift 289
AS 39, 46, 53, 62
ASC 49
asort 289
ASP 280, 310
asp 316
ASP.NET 280, 314, 364
asp_tags 286
assegnazione 286
ASSERTION 42
asserzioni 42, 97
associazione dati 262
associazioni 5, 83, 117
associazioni binarie 83
associazioni ISA 93
associazioni multiple 90, 101
associazioni n-arie 83
associazioni parziali 90
associazioni ricorsive 93, 120
associazioni ternarie 121
associazioni totali 90
associazioni univoche 90
atomicità 67
Attr 359
attributi 5, 17, 79, 116, 304, 343, 344, 349, 359
attributi composti o aggregati 79
attributi derivati 46, 80, 115, 212
attributi di sessione 281
attributi multivalore 80
attributi obbligatori 80
attributi opzionali o facoltativi 80
attributi semplici o atomici 79
AUTO_INCREMENT 165
AUTOCOMMIT 178
AutoExec 248
AutoGenerateColumns 329
AutoGenerateEditButton 329
AutoKeys 240
AutoPostBack 325
AVG 51, 162
azioni 237, 300
azioni attendibili 238

B

B+-tree 148
B-tree 147
back end 249
Backup 184
base di dati 3
BCNF 131, 135
bean 301
BEFORE 64
beforeFirst 271
BEGIN TRANSACTION 66
ben formato 339, 341
BETWEEN 48, 162
BETWEEN ... AND ... 41
BIGINT 160
BINARY 161
BindingNavigator 264
BindingSource 264, 265
BLOB 161
BOOLEAN 38, 160
Boolean 286
BoundField 329
buffer 144
business rule 97
Button 318, 319
BY 49, 51

C

C# 314
calcolo relazionale 6
Calendar 316, 321
CALL 177
CancelEvent 243
cancelRowUpdates 272
cardinalità 97
cardinalità degli attributi 80
cardinalità della partecipazione all'associazione 88
cardinalità della relazione 18
cardinalità di una associazione 86
cardinalità massima 88
cardinalità minima 88
CASCADE 41, 43
CASE 172
caselle combinate 228
caselle di riepilogo 228
caselle di testo associate 225
caselle di testo calcolate 228
catalogo 4
Catalogs 185
categoria 5
catene di overflow 146
CDATA 343
CEIL 162
CEILING 162
CGI 277, 278, 279
CHANGE 43
CHAR 38, 161
CHARACTER 38
CHECK 39, 41, 42, 165
CheckBox 318
CheckBoxList 318, 325
chiave 19
chiave candidata 19, 81
chiave di ricerca 145
chiave esterna 20
chiave multicampo 201, 202
chiave primaria 19, 81
chiavi esterne 166
Chiudi 238
choose 351
Class 267
classes 297
CLI 11
Close 312
CLR 314
coerenza 67
Collega campi master 230, 237
Collega campi secondari 230, 237
collegamenti estesi 345
collegamenti ipertestuali 339, 344
Collegamento ipertestuale 199
collisioni 146
Colonna associata 205
COLUMN 43
COM 258
ComboBox 264
Command 258
commenti 171, 286, 300, 342
COMMIT 178
commit 66
COMMIT WORK 66
CompareValidator 320
compilation 315
CONCAT 163
condivisione dei dati 3
condizione di ricerca 47
config 299, 301
config.inc.php 185
configuration 315
conflitto di impedenza 12
Connection 258, 267, 312, 313, 314
connessione 267
connessione a una origine dati 262
Consenti aggiunte 224
Consenti eliminazioni 224
Consenti filtri 224
Consenti lunghezza zero 199
Consenti modifiche 224
Consenti Visualizzazione 223
conservazione delle dipendenze 134
consistenza 67
CONSTRAINT 43
CONSTRAINTS 42
Container.DataItem 326
Contatore 198
contatore 165
Content-Length 277
Content-Type 277, 279
CONTENT_LENGTH 278, 279
contentType 362
contenuto informativo 5
Control 244, 245
controlli 224, 233, 316
controlli data bound 316, 325
controlli data source 316, 328
controlli di validazione 316, 320
controlli HTML server 316
controlli Web server 316
controllo dell'accesso 65
Controls 244, 245
ControlToCompare 320
ControlToValidate 320
ControlType 247
Cookie 303
cookie 277, 280, 292, 311
Cookies 311

Loescher Editore - Vietata la vendita e la diffusione

CopiaOggetto 238
COUNT 50, 162
Count 245
count 288
CREATE 268
CREATE ASSERTION 42
CREATE DATABASE 159
CREATE DOMAIN 39
CREATE FUNCTION 175
CREATE INDEX 43
CREATE PROCEDURE 176
CREATE ROLE 66
CREATE TABLE 40, 165
CREATE TEMPORARY TABLE 167
CREATE TRIGGER 64, 172
CREATE VIEW 62
createLSSerializer 360
CreateObject 312
createStatement 267, 271
cross-page 319
CSS 339, 340
CURDATE 163
cursore 12, 258, 268
cursore forward only 12
CURTIME 163
customErrors 315

D

dangling 25
DAO 259
data base gerarchico 7
data binding 262, 264, 265, 324
data-type 348
Data/ora 198
DataAdapter 259, 261, 323
database 3, 281, 293, 311
database a oggetti 8
database distribuiti 12
database relazionale 6
database reticolare 8
DatabaseMetaData 272
DataBind 324
DataBindings 367
DataFile 367
DataGrid 325, 327
DataGridView 265
DataItem 326
DataList 325, 326
DataMember 325, 367
DataReader 259, 260
DataSet 259, 261, 323, 367, 369
DataSource 265, 325
DataSourceId 325, 328
DataTable 259
DataTextField 325
DATE 38, 161
Date 212
DATE_ADD 163
DATE_FORMAT 163
DATE_SUB 163
DateAdd 212
DATEDIFF 163
DatePart 212
DATETIME 161
dati 3, 5
dati personali 424
dati sensibili 424
DAY 163
DAYNAME 163
DAYOFMONTH 163
DAYOFYEAR 163
DBA 12
DbCommand 259, 260, 322
DBEngine 259
DBMS 3, 281
DCL 10
DDL 10, 40
DDNS 283
debolmente tipizzato 286
debug 315
DEC 38
DECIMAL 38, 161
DECLARE 175
decomposizione senza perdita di informazioni 133
DEFAULT 40, 43, 44
defaultRedirect 315
DEFERRED 42
define 286
DELETE 41, 62, 64, 66, 262, 268
DELETE FROM 45
DeleteCommand 329
deleteRow 272
DELIMITER 173, 175, 176
Depth 367
DESC 49
DESCRIBE 159, 166
Descrizione 197
DetailsView 325, 328
diagramma degli stati 76
diagramma dei casi d'uso 76
diagramma dei componenti 76
diagramma della disposizione interna delle finestre 76
diagramma delle attività 76
diagramma di collaborazione 75
diagramma di navigazione 76
diagramma di sequenza 75
diagrammi di interazione tra oggetti 75
diagrammi di interfaccia 76
diagrammi entità/associazioni 75
diagrammi equivalenti 102
diagrammi ER 9, 75
diagrammi UML 9, 75
dichiarazione di tipo di documento 342
dichiarazione xml 341
dichiarazioni 300
Didascalia spostamento 223
differenza 23, 60
Dimensione campo 199
dipendenza funzionale 128
DirectoryIndex 284
diretta 83
direttive 300, 316
diritti di accesso 65
DisplayMode 320
DISTINCT 46, 50
DIV 162
dizionario dei dati 4
DLookup 213
DMCL 10
DML 10, 44
DMZ 283
DNS 283
DNS dinamico 283
DO 172
do while 287
DoCmd 243, 244
Document 358, 359
DOCUMENT_ROOT 278
DocumentBuilder 359
DocumentBuilderFactory 358, 359
documento XML 340
DocumentRoot 284
documentSource 365
DOM 358, 363, 369
DOMException 359
DomImplementation 359
DOMImplementationLS 359
dominio degli attributi 79
DOUBLE 161
Double 286
driver JDBC 266
driver JDBC-ODBC 266
driver ODBC 257, 281
DriverManager 267
DROP COLUMN 43
DROP CONSTRAINT 43
DROP DATABASE 159
DROP DOMAIN 39
DROP FUNCTION 175
DROP INDEX 44
DROP PROCEDURE 177
DROP TABLE 43
DROP TRIGGER 173
DROP VIEW 63
DropDownList 318, 325
DSN 257, 312
DTD 339, 342, 345, 346
dynaset 210

E

ECA 64
eccezione 305, 306
editor di query 180
editor di Visual Basic 242, 243
Element 359
EliminaOggetto 238
ELSE 172
EnableViewState 317
END 172
END TRANSACTION 66
ENGINE 165, 167
entità 78, 116
entità debole 81, 96
entità forte 96
ENUM 161
EOF 313
equi-join 26, 52, 54
ER 9, 75, 77
ereditarietà 93

Loescher Editore - Vietata la vendita e la diffusione

errori 298, 305, 306
errori tempo di controllo/tempo di utilizzo 331
ErrorMessage 320
errorPage 305
EseguiApplicazione 238
EseguiCodice 243
EseguiMacro 238
EseguiSQL 238
esemplare 5
Espansione immagine 223
esportazione di dati 182, 187, 195, 233
espressione di data binding 324
espressioni 300
estensione 5, 18, 78
Etichetta 199
etichette 228
Eval 326
eventi 64, 223, 240, 315, 316, 358
EXCEPT 60, 165
exception 305
Execute 312, 313, 314
execute 270
ExecuteNonQuery 260
executeQuery 267, 270
ExecuteReader 259, 260
executeUpdate 267, 268, 270
EXISTS 59
EXIT 157
EXP 162
expires 311
exploit 330
extension 284

F

fan-out 147
FIELDS 168
file 3, 4, 144
Fill 259, 261
Filtra al caricamento 223
filtri 207
Filtro 197, 223
finestra del database 196
FinestraMessaggio 238
first 271
FLOAT 38, 161
FLOOR 162
fogli stile 339
fogli stile XSL 347
foglio dati 203
FooterTemplate 326
for 287
FOR EACH ROW 64
for-each 348, 350, 352
foreach 288, 289
FOREIGN KEY 41, 166
foreign key 20
Form 244, 245, 311
form 278, 316
Form_ 243
forma normale di Boyce-Codd 131
Formato 199
formato dei comandi 38
Formatting 366
forme normali 127
Forms 244
FormView 325, 328
forName 267
forward 301
framework .NET 314
FROM 45, 66
From 277
front end 249
FTP 282, 283
FULL JOIN 55
full join 27
FUNCTION 175
function 288
funzione indirizzo 146
funzioni 162, 288
funzioni di aggregazione 49

G

Garante 425
generalizzazioni 93, 113
generalizzazioni esclusive o disgiunte 95
generalizzazioni parziali 94
generalizzazioni sovrapposte 95
generalizzazioni totali 94
gerarchie ISA 93
gestore dei metodi di accesso 144
gestore del buffer 144
gestore delle interrogazioni 144, 150
gestore dello spazio su disco 144
GET 278, 279, 290, 311
getAttributes 359
getChildNodes 359
getColumnCount 273
getColumnName 273
getColumnTypeName 273
getConnection 267
getDocumentElement 359
getDouble 268
getFeature 359
getInt 268
getMessage 305
getMetaData 273
getObject 268
getParameter 301
getParameterNames 301
getString 268
GetXML 367
giunzione 24
GNU GPL 155
grado di una associazione 83
grado di una relazione 18
grafo di definizione 8
GRANT 65, 178
GREATEST 164
GridView 325, 327
GROUP BY 51
gruppo macro 239

H

hash 146
HasRows 260
HAVING 215
HeaderTemplate 326
HeaderText 320
Health 184
HELP 157
HELP CONTENTS 158
hosting 282
HOUR 163
housing 282
href 344
HTTP 277, 280, 314
HTTP_USER_AGENT 278
httpd.conf 283
HyperLink 318

I

ID di sessione 281, 292, 311
identificatore di una entità 81
identificatore esterno 95, 116
identificatore interno 81
identificatore principale 116
identificatori 38, 160, 196
IF 164, 172
if 287, 351
If-Modified-Since 277
IIF 213
IIS 308
Image 318
Immagine 223
IMMEDIATE 42
impedence mismatch 12
Import 316
importazione di dati 187, 220
IN 41, 48, 162, 176
in_array 289
include 300, 301
inconsistenza 3, 66
inconsistenza dei dati 4
INDEX 43, 44
indice primario 145
indici 43, 145, 166, 202
Indicizzato 199, 202
indipendenza dei programmi 3, 10
indipendenza fisica 9
indipendenza logica 10
informazioni 3, 5
INNER JOIN 54
inner join 26, 54
InnoDB 165
INOUT 176
INSERT 62, 64, 65, 163, 268, 295, 313
INSERT INTO 44
insert row 272
InsertCommand 329
insertRow 272
INT 38, 160
INTEGER 38, 160
Integer 286
integrità 3
integrità di chiave 21
integrità referenziale 21, 41, 167, 208
intensione 5, 17
interfaccia CGI 277, 278, 279

Loescher Editore - Vietata la vendita e la diffusione

Internet Information Server 282, 308
interrogazioni 45
interrogazioni in comandi di manipolazione 62
interrogazioni nidificate 56
INTERSECT 60, 165
intersezione 23, 60
intestazioni 226, 234, 235
INTO 44, 47
INTO OUTFILE 169
INTO TABLE 168
inversa 83
IS NULL 48, 162
ISA 93
isAfterLast 271
isBeforeFirst 271
isErrorPage 305
isFirst 271
isLast 271
isolamento 67
IsPostBack 317
isSet 286
istanza 5, 17
istanza di associazione 86
istanza di una entità 78
istanza di una relazione 18
istruzioni di elaborazione 342
Item 245, 260
ItemTemplate 326

J

Java 266, 280, 297, 358
java.sql 267
JavaBeans 301
JavaScript 310
JavaServer Pages 280, 297
javax.xml.transform 361
JDBC 12, 257, 266, 267, 306
JOIN 54
join 24, 52, 150
join completo 25
join esterno 27
join interno 24, 26
join naturale 24, 52, 54
JSP 280, 297, 362
JSPContainer 297

K

ksort 289

L

Label 264, 318
LAMP 155
language 310
last 271
Last-Modified 277
LAST_INSERT_ID 165
layout di pagina 316
LCASE 163
LEAST 164
LEFT 163
Left 212
LEFT JOIN 55
left join 27
LENGTH 163
LIKE 49, 162
LIMIT 169
linee di join 217
LINES TERMINATED BY 168
linguaggio di markup 339
linguaggio dichiarativo 11
linguaggio navigazionale 11
linguaggio ospite 11
linguaggio per database 11
linguaggio SQL 37
ListBox 264, 318, 325
livelli di astrazione 9
LN 162
Load 316
LOAD DATA INFILE 168
LoadModule 284
LOCAL 168
LOCATE 164
locktype 314
LOG10 162
LOG2 162
Login 318
LONGBLOB 161
LONGTEXT 161
LOWER 163
LSSerializer 360

M

machine.config 315
macro 193, 229, 237
magic_quotes_gpc 330
markup 342
Maschera di input 199
maschere 193, 221
match 347
MathML 340
MAX 51, 162
MaximumValue 320
Me 245
mediazione incompleta 331
MEDIUMBLOB 161
MEDIUMINT 160
MEDIUMTEXT 161
Memo 198
merge scan 150
Messaggio errore 197, 199
metadati 4, 272
metalinguaggio 339
method 278, 347
Mid 212
MIN 51, 162
MinimumValue 320
MINUTE 163
modalità di accesso ai database 11
mode 315
modellazione 98
modelli 347
modelli di DBMS 5
modello di concorrenza 3
modello di dati 5
modello entità/associazioni 75
modello ER 75
modello gerarchico 7
modello relazionale 6, 17
modello reticolare 8
MODIFY 43
Module 245
moduli 193, 243
modulo 278, 302
molteplicità 87
molteplicità di una associazione 86
molteplicità massima 88
molteplicità minima 88
MONTH 163
MONTHNAME 163
motore di memorizzazione 165
motori di ricerca 282
MoveNext 313
moveToCurrentRow 272
moveToInsertRow 272
msxml 346
MyISAM 165
MyODBC 155
MySQL 155, 293
mysql 155, 157
MySQL Administrator 182
MySQL GUI Tools 179
MySQL Query Browser 179
mysql_affected_rows 296
mysql_close 294
mysql_connect 293
mysql_fetch_array 294
mysql_fetch_row 294
mysql_query 294, 295, 296
mysql_select_db 294
mysqld 155, 156

N

name 278
NamedNodeMap 358, 359
namespace 314, 343
NAT/PAT 283
NATURAL JOIN 54
nested loop 150
NEW 173
NEW ROW AS 64
newDocument 359
newDocumentBuilder 359
newInstance 359, 361
next 268, 271
NO ACTION 41
Node 358, 359
node 352
NodeList 358, 359
nodo 351
nodo di contesto 352
nome di dominio 282, 283
nomi degli attributi 79
nomi delle entità 78
non ridondanza 19
normalizzazione 127
NOT 48, 162
NOT NULL 41
notazione puntata 18, 310
NOW 163
Now 212
NSERT 262

Loescher Editore - Vietata la vendita e la diffusione

NULL 39, 44, 48, 162
null 17, 80
NUMERIC 38
Numerico 198
numero progressivo 81, 96
Nuovi valori 199

O

Object 286
occorrenza 17
ODBC 257, 281, 296, 311, 312
odbc_connect 296
odbc_exec 296
odbc_fetch_array 296
odbc_fetch_row 296
OdbcDataAdapter 261
Office 193
oggetti di Access 193, 244
oggetti predefiniti 301
oggetto 310
Oggetto OLE 198
OLD 173
OLD ROW AS 64
OLE DB 257, 258
OleDbCommand 260
OleDbDataAdapter 261
OMG 75
ON 41, 43, 54, 64, 65, 172, 178
ON DELETE 41
ON UPDATE 41
OODB 8
OODBMS 8
Open 312
OpenTable 243
Operator 320
operatori 162, 286
operatori logici 48
operatori relazionali 47
Option compare database 242
OR 48, 162
or 286
ORDBMS 8
order 348
ORDER BY 49
Ordina per 197
ordinamenti 49
ordine di tabulazione 224
organizzazione ad accesso calcolato 146
organizzazione fisica 145
organizzazione sequenziale 145
Origine controllo 226
Origine record 223, 232
Origine riga 205, 229
otherwise 351
ottimizzazione 150
OUT 176
out 299, 301
outer join 27

P

Page 316
page 299, 301, 304, 305, 362
page compilation 297
page translation 297, 300
PageContext 299
pageContext 301
pageOutput 315
PageSize 329
pannello dei comandi 248
paradigma Evento-Condizione-Azione 64
parametri 270, 288, 301
Parent 245
parentesi graffe 38, 349
parentesi quadre 38
parse 359
parser 339, 340, 346, 358
ParserConfigurationException 359
partial primary key 81
particolarizzazioni 93
partizionamento di entità 115
password 249
pattern 347
persistenza 67
PHP 280, 284
php.ini 284, 291, 292
PHP_SELF 291
phpinfo 284
phpMyAdmin 185
PI 162
piè di pagina 226, 234, 235
PK 20, 81
plugin 301
Popup 223
Port 284
Posizioni decimali 199
POST 278, 279, 290, 291, 311
postback 316, 317
PostBackUrl 319
posting cross-page 319
POW 162
POWER 162
PPK 81
predicati 97
prefisso 343
PreparedStatement 270
prepareStatement 270, 271
previous 271
PreviousControl 245
PreviousPage 319
prima forma normale 129
PRIMARY KEY 41
primary key 20, 81
Primi valori 211
principi di modellazione 98
principio di località dei dati 145
print 286, 298
printStackTrace 305
privilegi 65
PROCEDURE 176
procedure 65, 176
prodotto cartesiano 29
progettazione concettuale 9
progettazione di database 9
progettazione fisica 9, 149
progettazione logica 9, 113
proiezione 24, 46
prologo 341
Properties 244
proprietà 78
proprietà ACID 67
proprietà di partizionamento 94
provider OLE DB 259
provider SQL 259
PUBLIC 65
pulsanti di comando 229
pulsanti di opzione 229
pulsanti di spostamento 224
Pulsanti spostamento 223

Q

QBE 11, 188, 210, 211
QL 10
query 45, 193, 210, 347
query a campi incrociati 216
Query area 180
query correlate 59
query di comando 219
query di definizione dati 220
query nidificate 56
query parametriche 47, 171, 213
QUERY_STRING 278, 279
Querydef 259
QueryString 311
QUIT 157

R

radice 342, 347, 359
RadioButton 318
RadioButtonList 318, 325
raggruppamenti 51
RAND 162
RangeValidator 320
RBAC 66
RDBMS 6
Read 260
ReadXML 367
REAL 38
RecordSet 258, 312, 313, 314
Recordset 259
RecordsetClone 245
REFERENCES 41, 166
REFERENCING 64
referential integrity constraint 21
refreshRow 272
REGEXP 162
register_globals 291, 292
registrazione di un nome di dominio 282
registrazione nei motori di ricerca 282
regole attive 64
regole aziendali 97
regole di lettura delle associazioni 90
RegularExpressionValidator 320
relative 271
relazione 6, 17, 18
relazioni 208

Loescher Editore - Vietata la vendita e la diffusione

REMOTE_ADDR 278
REMOTE_HOST 278
RENAME TO 43
REPEAT 172
Repeater 325, 326
REPLACE 164
Report 244, 245
report 193, 232
Report_ 243
Reports 244
Request 310, 311
request 299, 301, 303
REQUEST_METHOD 278
RequiredFieldValidator 320
Response 310
response 299, 301, 302
Restore 185
RESTRICT 41, 43
restrizione 23
Result Area 181
ResultSet 267, 271, 272
resultSetConcurrency 271
ResultSetMetaData 272
resultSetType 271
RETURN 175
return 288
RETURNS 175
REVERSE 164
REVOKE 66
ricerca automatica 218
Ricerca guidata... 199
Richiesto 199
ridenominazione 23
ridondanza 3, 115
riferimenti a entità 343
RIGHT 164
Right 212
RIGHT JOIN 55
right join 27
Riquadro di spostamento 196, 248
riservatezza 3
ristrutturazione dello schema concettuale 113
rollback 66
ROLLBACK WORK 66
ROUND 162
runat 316
ruoli 66
ruoli delle entità 92

S

SAX 358, 363
scelta degli identificatori 116
schema 5
schema concettuale 75
schema di una relazione 17
schema esterno 9
schema fisico 9
schema logico 9, 113, 116
schema relazionale 6, 17, 116
Schemata 181
Screen 244, 245
script 286, 310, 316
scriptlet 300
SECOND 163
seconda forma normale 129
Section 244, 245
SELECT 45, 52, 65, 169, 260, 267, 294
select 348
SELECT INTO 47, 167, 170
SelectCommand 329
SelectedData 321
selettore di campo 203
selettore di record 203
selezione 23, 46
self-join 28, 53
SERIAL 161
Server 277, 310, 312
Server Connections 184
server FTP 283
Server Information 182
Server Logs 184
server side 277, 280
server Web 282, 283, 308
SERVER_NAME 278
SERVER_PORT 278
SERVER_SOFTWARE 278
Service Control 183
Servizi 156
servizio Web 282
servlet 280, 297, 298, 300
Session 310, 311
session 299, 301, 304
Session_id 292
session_start 292
sessioni 280, 281, 292, 311
SET 44, 161, 170
Set 247
SET CONSTRAINTS 42
SET DEFAULT 41
SET NULL 41, 43
SET ROLE 66
Set-Cookie 277, 280
Setcookie 292
setDouble 270
setInt 270
setObject 270
setString 270
sezione di codice 316
sezione di layout 316
sezioni 226, 234
SGML 339
short_open_tag 286
show 344
SHOW CREATE TABLE 165
SHOW DATABASES 159
SHOW TABLES 159
ShowMessageBox 320
ShowSummary 320
Sì/No 198
significato estensionale 5, 18
significato intensionale 5, 17
sinonimi 146
sistema informatico 3
sistema informativo 3
SMALLINT 38, 160
SMIL 340
SOAP 314, 340
sort 289, 348
sort-merge scan 150
sottomaschere 229
sottoquery 56, 353
sottoreport 237
sottoschema 9
SOURCE 159, 171
spazio dei nomi 343
specializzazioni 93
SQL 11, 37, 188, 214, 257, 260, 267, 268, 270, 281, 294, 295, 296, 312, 313, 314
SQL dinamico 11
SQL embedded 11
SQL injection 330, 331
SQL-2 37, 54, 65
SQL-3 37, 66
SQL-86 37
SQL-89 37
SQL:1999 37
SQL:2003 37
SqlDataAdapter 261
SqlDataSource 328
SQLException 306
SQRT 162
Stampa 238
standard input 279
standard output 279
START TRANSACTION 178
Startup Variables 183
Statement 267, 268
storage engine 165
stored procedure 11, 65
stored routine 175
STR_TO_DATE 163
STRCMP 164
String 286
stringhe 289
strlen 289
strtolower 289
strtoupper 289
struttura key-sequenced 148
strutture ad albero 147
strutture primarie 145
strutture secondarie 145
stylesheet 347
submit 278
SUBSTR 164
SUM 50, 162
Sum 213
superchiave 19
SVG 340
switch 287
Switchboard Items 248
SYSDATE 163
System.Data.Odbc 259
System.Data.OleDb 259
System.Data.SqlClient 259
system.web 315
System.Xml 365, 366

T

tabella di verità degli operatori booleani 48

Loescher Editore - Vietata la vendita e la diffusione

tabelle 6, 17, 193, 197
tabelle temporanee 47, 58, 63
TABLE 40, 43
Table editor 181
TableAdapter 264, 265
Tabledef 259
tag 339
TCP 277
TEE 159
template 325, 326, 347, 352
terza forma normale 130
test 351
Testo 198
TEXT 161
Text 320
text 352
TextBox 264, 318
TextField 367
THEN 172
theta-join 29, 56
TIME 38, 161
TIMESTAMP 161
TINYBLOB 161
TINYINT 160
TINYTEXT 161
Tipo immagine 223
Tipo origine riga 205, 229
TO 65, 178
Tomcat 296
trace 315
trace.axd 315
TRANSACTION 66, 178
transazioni 66, 178
transform 362
Transformer 361
TransformerFactory 361
transformSource 365
TrasferisciDatabase 238
trasformazione 347, 348
trattamento dei dati personali 424
TreeNodeBinding 367
TreeView 367
TRIGGER 64
trigger 64, 172
TRIM 164
trim 289
TrovaRecord 238
TrovaSuccessivo 238
TRUNCATE 162
tupla 17
tuple dangling 25
TYPE 167
Type 320
type 344
TypeOf 247

U

uasort 289
UCASE 164
uksort 289
UML 9, 75, 77
UNION 60, 165
unione 23, 60
UNIQUE 41, 43
univocità 19
UNSIGNED 160
UNTIL 172
UPDATE 41, 44, 62, 64, 66, 262, 268, 296, 314
Update 262, 313, 314
UpdateCommand 329
updateDouble 272
updateInt 272
updateObject 272
updateRow 272
updateString 272
UPPER 164
URI 343, 345
URL 343
URL codificati 281
URN 343
USE 159
useBean 301
User Administration 183
USING 54
usort 289
utenti di database 12
UTF-8 341

V

VaiAControllo 238
VaiAPagina 238
VaiARecord 238
Validation Service 346
ValidationExpression 320
ValidationSummary 320
validazione 346
validazione dell'input 330
valido 339, 345
Valido se 197, 199
Valore predefinito 199
Valori univoci 211
VALUE BETWEEN ... AND ... 41
VALUE IN 41
value-of 348, 350
VALUES 44
ValueToCompare 320
Valuta 198
vantaggi dei database 3
VARBINARY 161
VARCHAR 161
variabili 170
variabili d'ambiente 278
VBA 241
VBScript 310
VIEW 62, 63
ViewState 317
vincoli 5
vincoli di chiave 21
vincoli di chiave primaria 97
vincoli di integrità 21, 97
vincoli di tupla 21
vincoli espliciti 97
vincoli impliciti 97
vincoli interrelazionali 21
vincoli intrarelazionali 21, 165
vincoli intrarelazionali predefiniti 41
vincoli referenziali 97
vincolo di dominio 21
vincolo di integrità referenziale 21, 41
vincolo su valori 21
VisibleDate 321
viste 9, 31, 171, 218
viste logiche 62, 63, 66
viste materializzate 31
viste virtuali 31
Visual Basic 259, 314
Visualizzazione predefinita 222
volume dei dati 113
vulnerabilità 330

W

WAMP 155
WAP 340
Web service 314
WEB-INF 297
web.config 315
web.xml 297
WHEN 64, 172
when 351
WHERE 44, 45
WHILE 172
while 287
WITH CHECK OPTION 171
WML 340
Workspace 259
write 310
WriteElementString 366
WriteEndDocument 366
WriteEndElement 366
WriteRaw 366
WriteStartDocument 366
WriteStartElement 366
writeToString 360
WriteXML 367

X

X3D 340
Xampp 284
XHTML 340, 346
XLink 339, 340, 344
XML 12, 259, 314, 339, 358, 362, 364
Xml 316, 364
xml-stylesheet 347
XmlDataDocument 369
XmlDataSource 328, 367
XmlDocument 369
XmlNode 369
xmlns 343
XmlTextReader 365
XmlTextWriter 366
XPath 351
XPointer 345
XSL 339, 340, 347, 364
xsl:template 347
XSLT 347

Y

YEAR 161, 163

Loescher Editore - Vietata la vendita e la diffusione

QUESTO VOLUME, SPROVVISTO DEL TALLONCINO A FRONTE (O OPPORTUNAMENTE PUNZONATO O ALTRIMENTI CONTRASSEGNATO), È DA CONSIDERARSI COPIA DI **SAGGIO - CAMPIONE GRATUITO**, FUORI COMMERCIO (VENDITA E ALTRI ATTI DI DISPOSIZIONE VIETATI: ART. 17, C. 2 L. 633/1941). ESENTE DA IVA (DPR 26.10.1972, N. 633, ART. 2, LETT. D). ESENTE DA DOCUMENTO DI TRASPORTO (DPR 26.10.1972, N. 633, ART. 74).

ISBN 978-88-8433-027-7
SCORZONI INFO ITC DATABASE
1027

In copertina: *Strumenti flessibili.* © ICPonline

Informatica
DataBase

Questo testo è destinato alla materia **Informatica** delle classi quarte e quinte degli ITIS a indirizzo informatico (Abacus e di ordinamento) e degli ITC a indirizzo programmatori (Mercurio e di ordinamento).

La prima parte del volume, dopo una breve introduzione generale, tratta gli argomenti teorici relativi ai **database relazionali**, al linguaggio **SQL** e alla **progettazione di database** (concettuale, logica e fisica).
La spiegazione del modello relazionale e del linguaggio SQL è stata anticipata rispetto agli argomenti di progettazione in modo da consentire in breve tempo di comprendere gli argomenti relativi all'utilizzo di DBMS e alla realizzazione di applicazioni (per chi desidera partire dalla progettazione dei database è sufficiente però seguire i capitoli in ordine diverso).
La seconda parte è più pratica e tratta: l'utilizzo degli ambienti DBMS personali più diffusi (**MySQL** e **Access**); le modalità di **accesso ai database** dalle applicazioni, in particolare utilizzando i linguaggi **Visual Basic** e **Java**; le modalità di accesso ai database in rete via Web con le **tecnologie server side** più diffuse (**PHP**, **JSP**, **ASP** e **ASP.NET**); il linguaggio **XML** e l'accesso a documenti XML da applicazioni stand alone e server side.
L'ultima parte presenta i **temi d'esame** proposti negli ultimi anni agli Esami di Stato degli ITIS a indirizzo informatico (Abacus e di ordinamento) e degli ITC a indirizzo programmatori (Mercurio e di ordinamento) con **esempi di svolgimento**.

Per tutti gli argomenti affrontati si è cercato di dare una interpretazione molto attuale, non soltanto dal punto di vista teorico ma anche con riferimenti concreti agli strumenti utilizzati nella pratica lavorativa.
Il libro è diviso in **moduli**, a loro volta ripartiti in **capitoli**.

- **Modulo 1** – I database
  1 - Introduzione ai database
  2 - Database relazionali
  3 - SQL
- **Modulo 2** – La progettazione di database
  4 - Il modello concettuale
  5 - Il modello logico
  6 - Il modello fisico
- **Modulo 3** – DBMS
  7 - MySQL
  8 - Access
- **Modulo 4** – Accesso a database
  9 - Accesso ai database dalle applicazioni
  10 - Accesso a database via Web
- **Modulo 5** – XML
  11 - XML e XSL
  12 - Elaborazione di documenti XML
- **Modulo 6** – Progetti
  13 - Temi d'esame
  14 - Esempi di svolgimento
- **Appendice - Database e diritto**

Per consentire un utilizzo più agevole dei volumi, che costituiscono l'opera, ogni capitolo comprende una sezione di **teoria** corredata di numerosi esempi, e una sezione di **verifica** con domande, test, esercizi e proposte di lavoro. Gli argomenti per l'effettuazione di esercitazioni di **laboratorio** sono integrati strettamente nel testo.

QUESTO CORSO È COSTITUITO DA:
ISBN 978-88-8433-**025**-3 IL COMPUTER E LA PROGRAMMAZIONE IN VISUALBASIC
ISBN 978-88-8433-**026**-0 IL COMPUTER E LA PROGRAMMAZIONE IN PASCAL
ISBN 978-88-8433-**630**-9 SISTEMI OPERATIVI, RETI E INTERNET PER L'AZIENDA
ISBN 978-88-8433-**031**-4 CD-ROM PER L'INSEGNANTE
ISBN 978-88-8433-**027**-7 DATABASE
ISBN 978-88-8433-**029**-1 PROGRAMMAZIONE IN C ++
ISBN 978-88-8433-**030**-7 PROGRAMMAZIONE IN JAVA

J027
SCORZONI
INFORMATICA
DATABASE

EDUCAZIONE TECNICA SUPERIORE