

5° anno

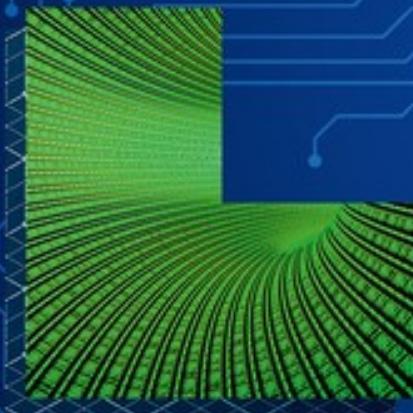
CESARE
IACOBELLI

MARIALAURA
AJME

VELIA
MARRONE



EPROGRAM



ISTITUTI TECNICI
SETTORE TECNOLOGICO
ARTICOLAZIONE INFORMATICA

PROGETTAZIONE
DI DATABASE E
LINGUAGGIO SQL

PROGRAMMAZIONE LATO
CLIENT E LATO SERVER

NUOVO ESAME DI STATO

Cesare Iacobelli Marialaura Ajme Velia Marrone

EPROGRAM

5° ANNO

ISTITUTI TECNICI - SETTORE TECNOLOGICO
ARTICOLAZIONE INFORMATICA



www.mondadorieducation.it

Questo ebook contiene materiale protetto da copyright e non può essere copiato, riprodotto, trasferito, distribuito, noleggiato, licenziato o trasmesso in pubblico, o utilizzato in alcun altro modo ad eccezione di quanto è stato specificamente autorizzato dall'editore, ai termini e alle condizioni alle quali è stato acquistato o da quanto esplicitamente previsto dalla legge applicabile. Qualsiasi distribuzione o fruizione non autorizzata di questo testo così come l'alterazione delle informazioni elettroniche sul regime dei diritti costituisce una violazione dei diritti dell'editore e dell'autore e sarà sanzionata civilmente e penalmente secondo quanto previsto dalla Legge 633/1941 e successive modifiche.

Questo ebook non potrà in alcun modo essere oggetto di scambio, commercio, prestito, rivendita, acquisto rateale o altrimenti diffuso senza il preventivo consenso scritto dell'editore. In caso di consenso, tale ebook non potrà avere alcuna forma diversa da quella in cui l'opera è stata pubblicata e le condizioni incluse alla presente dovranno essere imposte anche al fruitore successivo.

<i>Redazione</i>	Jennifer Santoro - (duDAT S.r.l. Bologna)
<i>Progetto grafico</i>	duDAT S.r.l. (Bologna)
<i>Impaginazione</i>	duDAT S.r.l. (Bologna)
<i>Direzione artistica sistema visivo delle copertine</i>	46xy studio
<i>Realizzazione della copertina</i>	Plum
<i>Disegni</i>	duDAT S.r.l. (Bologna)
<i>Ricerca iconografica</i>	duDAT S.r.l. (Bologna)

In copertina:
Seamless Block Data Connection Network Background © Gettyimages /
Binary Code Background © Gettyimages / Concept of data streaming ©
Gettyimages / Tunnel digitale © Gettyimages / Circuit board background.
Vector electronic background © Shutterstock

<i>Contenuti digitali</i>	
<i>Progettazione</i>	Fabio Ferri, Lorenzo Testa
<i>Redazione</i>	duDAT S.r.l. (esercizi commentati, laboratori "case studies", lezioni LIM)
<i>Realizzazione</i>	duDAT S.r.l. (esercizi commentati, laboratori "case studies", lezioni LIM), IMMAGINA S.r.l. (audio), Lumina Datamatics (mappe modificabili)

Alla data di stampa dell'opera non sono state pubblicate dal MIUR le simulazioni della seconda prova scritta dell'Esame di Stato dell'indirizzo Informatica e Telecomunicazioni, Articolazione Informatica. Forniremo tracce di seconde prove scritte nella sezione Contenuti Digitali Integrativi Docente, non appena il MIUR avrà reso pubbliche le simulazioni.



FONT biancoenero®

Questo libro usa la font ad Alta Leggibilità biancoenero®
di biancoenero edizioni, disegnata da Umberto Mischi.
La font è gratuita per studenti e insegnanti.

Avvertenza: Occasionalmente, possono essere visibili in questo testo nomi, confezioni e marchi commerciali di prodotti o società. Non li abbiamo eliminati per non rendere le esemplificazioni e le immagini irreali e "false", quindi didatticamente inefficaci. L'autore e l'editore non intendono sostenere che i prodotti fotografati o citati siano migliori o peggiori di altri, né indirettamente consigliarne o sconsigliarne l'acquisto: non esiste alcun rapporto di nessun genere con i relativi produttori.

L'editore fornisce - per il tramite dei testi scolastici da esso pubblicati e attraverso i relativi supporti - link a siti di terze parti esclusivamente per fini didattici o perché indicati e consigliati da altri siti istituzionali. Pertanto l'editore non è responsabile, neppure indirettamente, del contenuto e delle immagini riprodotte su tali siti in data successiva a quella della pubblicazione, distribuzione e/o ristampa del presente testo scolastico.

Per eventuali e comunque non volute omissioni e per gli aventi diritto tutelati dalla legge, l'editore dichiara la piena disponibilità.

La realizzazione di un libro scolastico è un'attività complessa che comporta controlli di varia natura. Essi riguardano sia la correttezza dei contenuti che la coerenza tra testo, immagini, strumenti di esercitazione e applicazioni digitali. È pertanto possibile che, dopo la pubblicazione, siano riscontrabili errori e imprecisioni. Mondadori Education ringrazia fin da ora chi vorrà segnalarli a:

Servizio Clienti Mondadori Education
e-mail servizioclienti.edu@mondadorieducation.it
numero verde **800 123 931**

PRESENTAZIONE

STRUTTURA DEL CORSO

Il testo è organizzato in **lezioni** che si sviluppano per piccoli blocchi e la trattazione è arricchita da numerosissimi schemi e grafici. In questo modo si facilitano lo studio e i percorsi di apprendimento. Ogni lezione ha una sua organizzazione indipendente, in modo che sia possibile saltare alcune lezioni ritenute troppo approfondite oppure organizzare la sequenza in modo differente.

Alla fine di ogni lezione viene proposta una **verifica delle conoscenze** relative all'argomento trattato.

L'insieme delle lezioni sullo stesso argomento è organizzato in **unità di apprendimento**.

Le unità 6 e 7 possono essere considerate alternative: trattano gli stessi temi, cioè la realizzazione di programmi lato server che si possono interfacciare con un database. Ciò che cambia è l'ambito operativo: nell'unità 6 si opera in un ambiente Open Source con **PHP** e **MySQL**, nell'unità 7 si utilizzano **VB.NET** e **ASP.NET**.

La conclusione di ogni unità è costituita da:

- una **sintesi articolata per lezione** (*Ripassiamo insieme*), affiancata da una **mappa concettuale**; sia la sintesi sia la mappa sono pensate in ottica inclusiva e sono formulate in caratteri ad alta leggibilità (la mappa è disponibile anche in formato digitale modificabile);
- una sezione per la **verifica delle conoscenze**, costituita di un **test strutturato** (disponibile anche in versione digitale autocorrettiva) e di un insieme di **domande aperte** (*Preparati per il colloquio orale*). Le domande sono pensate sia per l'interrogazione, il ripasso e la verifica *in itinere*, sia per la preparazione al colloquio orale dell'esame di Stato;
- una sezione *In English, please*, che contiene una sintesi dei contenuti dell'unità (*Abstract*), esercizi, un glossario su carta dei termini più usati, un glossario interattivo in italiano e in inglese con funzione di ricerca e un file audio per la pronuncia del testo, tutto in lingua inglese;
- un momento di preparazione alla **seconda prova scritta** dell'esame di Stato (*Preparati all'esame di Stato*), con l'estratto svolto di un tema d'esame assegnato in passato.

Molto ampio è l'**apparato delle esercitazioni**, che presenta diverse tipologie di esercizi, **esempi** e **laboratori**. Alla fine del volume vengono proposti **esercizi di riepilogo** (*Esercizi finali*), divisi per argomento.

Il testo prevede la codifica sempre in due linguaggi (**PHP** e **MySQL** e **Visual Basic.NET** e **ASP.NET**), rendendo così possibile per il docente scegliere quale percorso seguire, usando uno solo dei linguaggi, o usandoli in alcuni casi entrambi per evidenziarne le differenze.

IL NUOVO ESAME DI STATO

Alla seconda prova scritta dell'esame di Stato abbiamo dedicato schede a fine unità e una sezione a fine volume. Nel formularle abbiamo tenuto conto dei recenti *Quadri di riferimento per la redazione e lo svolgimento della seconda prova scritta dell'esame di Stato*, evidenziando i **nuclei tematici fondamentali** e gli **obiettivi**.

Abbiamo proposto allo studente anche un esempio di presentazione di un progetto annuale di un prodotto realizzato durante l'ultimo anno di studi, presentabile al colloquio orale dell'esame di Stato. Si tratta della progettazione di una **App per ipovedenti per smartphone**, un lavoro interdisciplinare realmente effettuato da studenti del quinto anno.

ALTERNANZA SCUOLA-LAVORO. PERCORSI PER LE COMPETENZE TRASVERSALI E PER L'ORIENTAMENTO (PCTO)

Abbiamo proposto, per i *Percorsi per le competenze trasversali e per l'orientamento*, una sezione che guida a compiere una sintesi dell'esperienza, incentrata su una riflessione sui percorsi per le competenze trasversali attuati e su una relazione finale d'esempio.

Si desidera ringraziare tutte le persone che con suggerimenti di correzioni e modifiche ci hanno permesso di rendere migliore e più attuale il testo. Rivolgiamo un ringraziamento particolare a Raffaella Romeo, che ha formulato le mappe concettuali poste alla fine delle unità, a Federico Brignone per l'aiuto dato ad affrontare i Big Data, a Stefano Gioda per l'apporto dato alla realizzazione della relazione e presentazione per l'esame di Stato e a Sara Iacobelli per l'aiuto nella revisione degli esercizi.

Gli autori

SOMMARIO

UNITÀ 1

BASI DI DATI

PREREQUISITI, PER COMINCIARE CONOSCENZE, ABILITÀ, COMPETENZE	2
1 I dati in azienda	3
2 Memorizzare dati	8
3 I file ad accesso diretto	13
4 Dal filesystem alle basi di dati	17
5 Architettura	21
6 Linguaggi e utenti	25
7 Sicurezza nelle basi di dati	28
RIPASSIAMO INSIEME	34
MAPPA CONCETTUALE	35
AUTOVALUTAZIONE DELLE CONOSCENZE	36
SVILUPPO DELLE COMPETENZE	
• CLIL – In English, please	37

UNITÀ 2

PROGETTARE UNA BASE DI DATI

PREREQUISITI, PER COMINCIARE CONOSCENZE, ABILITÀ, COMPETENZE	38
1 La progettazione di un database	39
2 Il modello E/R - Entità e attributi	41
3 Le chiavi	44
4 Le relazioni 1:1 e 1:N	45
5 Le associazioni N:N e le relazioni con attributi	48
6 Le associazioni binarie, unarie e multiple	51
7 Entità deboli con identificazione esterna. Gerarchie	53
8 Schemi e sottoschemi	56
9 Progettare un database	58
RIPASSIAMO INSIEME	62
MAPPA CONCETTUALE	63
AUTOVALUTAZIONE DELLE CONOSCENZE	64
SVILUPPO DELLE COMPETENZE	
• CLIL – In English, please	65
PREPARATI ALL'ESAME DI STATO	
• Estratto risolto di una seconda prova scritta assegnata in passato	66

UNITÀ 3

MODELLO RELAZIONALE

PREREQUISITI, PER COMINCIARE CONOSCENZE, ABILITÀ, COMPETENZE DISCIPLINARI	72
--	-----------

CONTENUTI DIGITALI INTEGRATIVI



- ESERCIZI COMMENTATI**
Segui la risoluzione passo passo
- GLOSSARIO CLIL**
- MAPPA MODIFICABILE**
- TEST**
Svolgi il test interattivo
- AUDIO**
Ascolta la pronuncia del testo

- ESERCIZI COMMENTATI**
Segui la risoluzione passo passo
- GLOSSARIO CLIL**
- MAPPA MODIFICABILE**
- TEST**
Svolgi il test interattivo
- AUDIO**
Ascolta la pronuncia del testo

- ESERCIZI COMMENTATI**
Segui la risoluzione passo passo
- LABORATORI CASE STUDIES**
Approfondisci con i casi pratici

CONTENUTI DIGITALI INTEGRATIVI



1 I modelli logici	73
2 Il modello relazionale	78
3 Ristrutturazione dello schema E/R	85
4 Traduzione nel modello logico	88
5 Operazioni sulle tabelle relazionali	94
6 Algebra relazionale	102
7 Normalizzazione	104
8 Vincoli di integrità referenziale	110
RIPASSIAMO INSIEME	112
MAPPA CONCETTUALE	113
AUTOVALUTAZIONE DELLE CONOSCENZE	114
SVILUPPO DELLE COMPETENZE	
• CLIL – In English, please	115
PREPARATI ALL'ESAME DI STATO	
• Estratto risolto di una seconda prova scritta assegnata in passato	116

UNITÀ 4**IL LINGUAGGIO SQL**PREREQUISITI, PER COMINCIARE
CONOSCENZE, ABILITÀ, COMPETENZE

1 Definire lo schema	121
2 Modificare lo schema di una base di dati	130
3 Modificare i dati	132
4 L'istruzione SELECT	136
5 Altri usi dell'istruzione SELECT	142
6 L'operazione JOIN	144
7 Tipi di JOIN	149
8 Funzioni di aggregazione	153
9 Raggruppamenti	157
10 Query complesse	161
11 Subquery complesse	164
12 Unione, intersezione e differenza	169
13 Le viste	172
14 Sicurezza dei dati	174
15 Le transazioni	179
RIPASSIAMO INSIEME	183
MAPPA CONCETTUALE	185
AUTOVALUTAZIONE DELLE CONOSCENZE	186
SVILUPPO DELLE COMPETENZE	
• CLIL – In English, please	187
PREPARATI ALL'ESAME DI STATO	
• Estratto risolto di una seconda prova scritta assegnata in passato	188

- ESERCIZI COMMENTATI**
Segui la risoluzione passo passo
- LABORATORI CASE STUDIES**
Approfondisci con i casi pratici
- GLOSSARIO CLIL**
- TEST**
Svolgi il test interattivo
- AUDIO**
Ascolta la pronuncia del testo

UNITÀ 5**PROGRAMMARE IN RETE**

**PREREQUISITI, PER COMINCIARE
CONOSCENZE, ABILITÀ, COMPETENZE**

1 Programmare applicazioni web	193
2 Programmare lato Client	197
3 Programmare lato Server	199
RIPASSIAMO INSIEME	202
MAPPA CONCETTUALE	203
AUTOVALUTAZIONE DELLE CONOSCENZE	204
SVILUPPO DELLE COMPETENZE	
• CLIL – In English, please	205

UNITÀ 6**PHP E MYSQL**

**PREREQUISITI, PER COMINCIARE
CONOSCENZE, ABILITÀ, COMPETENZE**

1 Linguaggio PHP	207
2 HTML e PHP	213
3 Passaggio di parametri in PHP	215
4 Connessione al db e visualizzazione dati	219
5 Inserimento e modifica dati	224
6 Login	226
7 Importare ed esportare dati	229
8 Esercizio completo in PHP	232
RIPASSIAMO INSIEME	234
MAPPA CONCETTUALE	235
AUTOVALUTAZIONE DELLE CONOSCENZE	236
SVILUPPO DELLE COMPETENZE	
• CLIL – In English, please	237
PREPARATI ALL'ESAME DI STATO	
• Estratto risolto di una seconda prova scritta assegnata in passato	238

UNITÀ 7**DATABASE IN RETE - ASP.NET**

**PREREQUISITI, PER COMINCIARE
CONOSCENZE, ABILITÀ, COMPETENZE**

1 HTML e ASP.NET	243
2 Passaggio parametri	250
3 Connessione al database	255
4 Visualizzazione dati	259
5 Inserimento e modifica dati	266
6 Login	269
7 Importare ed esportare dati	272
8 Esercizio completo IN ASP.NET	278
RIPASSIAMO INSIEME	282

CONTENUTI DIGITALI INTEGRATIVI**ESERCIZI COMMENTATI**

Segui la risoluzione passo passo

LABORATORI CASE STUDIES

Approfondisci con i casi pratici

GLOSSARIO CLIL**MAPPA MODIFICABILE****TEST**

Svolgi il test interattivo

AUDIO

Ascolta la pronuncia del testo

LABORATORI CASE STUDIES

Approfondisci con i casi pratici

GLOSSARIO CLIL**MAPPA MODIFICABILE****TEST**

Svolgi il test interattivo

AUDIO

Ascolta la pronuncia del testo

ESERCIZI COMMENTATI

Segui la risoluzione passo passo

LABORATORI CASE STUDIES

Approfondisci con i casi pratici

GLOSSARIO CLIL**MAPPA MODIFICABILE****TEST**

Svolgi il test interattivo

CONTENUTI DIGITALI INTEGRATIVI



MAPPA CONCETTUALE	283
AUTOVALUTAZIONE DELLE CONOSCENZE	284
SVILUPPO DELLE COMPETENZE	
• CLIL – In English, please	285
PREPARATI ALL'ESAME DI STATO	
• Estratto risolto di una seconda prova scritta assegnata in passato	286
<hr/>	
UNITÀ 8	
BIG DATA E SISTEMI NOREL	
PREREQUISITI, PER COMINCIARE CONOSCENZE, ABILITÀ, COMPETENZE	292
1 L'approccio "Cloud"	293
2 Big Data	295
3 Database non relazionali	297
4 Gli Open Data	302
RIPASSIAMO INSIEME	308
MAPPA CONCETTUALE	309
AUTOVALUTAZIONE DELLE CONOSCENZE	310
SVILUPPO DELLE COMPETENZE	
• CLIL – In English, please	311
<hr/>	
ASL - I PERCORSI PER LE COMPETENZE TRASVERSALI E PER L'ORIENTAMENTO	
1 I percorsi per le competenze trasversali e per l'orientamento	314
2 Riconoscere e valutare le competenze trasversali	318
3 Esempio di una relazione conclusiva sui Percorsi	321
<hr/>	
RELAZIONE E PRESENTAZIONE DI UN PROGETTO PER IL COLLOQUIO ORALE DELL'ESAME DI STATO	
1 Redigere la relazione di un progetto: un esempio	331
2 Presentare un progetto: un esempio	337
<hr/>	
PREPARAZIONE ALLA SECONDA PROVA SCRITTA DELL'ESAME DI STATO	341
<hr/>	
ESERCIZI FINALI	370
<hr/>	
APPENDICI	
1 XAMPP e MySQL	381
2 Visual Studio e SQL Server	386
3 Creare pagine ASPX	391
4 I comandi SQL	395
<hr/>	
SOLUZIONI DEI TEST	397
<hr/>	
INDICE ANALITICO	398



hub



HUBSCUOLA.IT

L'AMBIENTE INTERATTIVO PER LA DIDATTICA DIGITALE.



hub
SCUOLA

HUB Scuola è il centro del sistema per la didattica digitale. La piattaforma digitale per lo studio e l'insegnamento che permette al docente e allo studente di fruire del libro digitale e delle risorse multimediali integrate nel libro; consente inoltre di condividere i contenuti multimediali disponibili sulla piattaforma.



hub
YOUNG

HUB Young è la App per usare la versione digitale del libro di testo. Il libro diventa pratico e intuitivo, stimola le dinamiche di apprendimento, favorisce l'inclusione e potenzia i risultati individuali.



hub
KIT

I contenuti digitali integrativi che arricchiscono ed espandono il libro di testo sono raccolti in HUB Kit: audio, video, esercizi interattivi, materiali aggiuntivi e contenuti scaricabili, mappe concettuali, laboratori digitali e gallerie d'immagini.



hub
SMART

*NOVITÀ

HUB Smart è l'App per guardare i video e ascoltare gli audio del libro di testo, oltre a consentire agli studenti di allenarsi con i Test, direttamente dallo smartphone.



hub
TEST

*NOVITÀ

HUB Test è la piattaforma per docenti e studenti, per creare verifiche e mettersi alla prova. Contiene un ricco database di quesiti disponibili ed è utilissima anche per l'allenamento individuale.



hub
CAMPUS

HUB Campus è il portale disciplinare ricco di risorse per il docente.

Come accedere a HUB Scuola?



Vai su www.hubscuola.it e accedi con le tue credenziali Mondadori Education. Non sei registrato?

Vai su www.hubscuola.it, clicca su "registrazione" e compila il form.

Scarica HUB Young



Scarica l'App da HUB Scuola o dai principali store online per fruire del tuo libro e di tutte le risorse multimediali integrate.

Lancia l'App, effettua il login e nella libreria troverai tutti i libri che hai attivato.



Video, Audio e Test sempre a portata di mano.

HUB Smart è l'App gratuita che permette di fruire di molti contenuti digitali del libro di testo direttamente dallo smartphone e dal tablet, senza necessità di registrazione.

• HUB Smart è intuitiva:

l'applicazione rende la didattica digitale alla portata di tutti.

• Usare HUB Smart è facile:

verifica che sulla quarta di copertina del tuo libro di testo ci sia l'icona HUB Smart. Scarica l'App gratuita dai principali store online.

• Lancia HUB Smart

e clicca sullo schermo per attivare la videocamera. Puoi inquadrare il QRCode presente nelle pagine del libro e fruire del contenuto digitale a disposizione.

Se i contenuti digitali del tuo libro sono molti, troverai un unico QRCode nelle pagine di apertura dell'unità: inquadrandolo, vedrai l'elenco dei video, audio e Test dell'unità e potrai fruirne da subito.



I contenuti digitali sono sempre a disposizione nella Cronologia e puoi salvarli tra i Preferiti.



HUB Test è la nuova piattaforma semplice e intuitiva che consente al docente di creare verifiche personalizzate per tutte le esigenze della classe. HUB Test permette di costruire verifiche scegliendo l'ordine scolastico, la materia e l'argomento.

Con HUB Test puoi:

- scegliere un Test tra le moltissime verifiche già pronte,
- creare Test da zero scegliendo tra quesiti di varia tipologia: risposta multipla, vero/falso, completamento, trova l'errore, raggruppamento e risposta aperta,
- utilizzare singoli quesiti per verifiche personalizzate,
- infine, ritrovare sempre i test salvati nel tuo archivio.



La piattaforma permette di generare automaticamente le verifiche, fino a tre versioni differenti per file diverse. La verifica è pronta: può essere assegnata alla Classe Virtuale, oppure stampata e consegnata agli studenti.



HUB Test è una risorsa validissima anche per gli studenti che possono allenarsi nelle varie materie e tenere traccia dei propri progressi. Il sistema infatti restituisce il Test corretto con feedback.

INVALSI

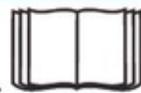


L'ambiente in cui lo studente può prepararsi alle prove ufficiali: è creato per avvicinare l'alunno all'interfaccia della piattaforma INVALSI e gli consente di svolgere le prove in modalità Computer Based. Su HUB Scuola sono a disposizione esercitazioni in italiano, matematica e inglese come previsto dalla normativa INVALSI.

LINK UTILI

- La piattaforma per la didattica digitale: hubscuola.it
- Il portale con le nostre novità: mondadorieducation.it
- L'assistenza per tutti: assistenza.hubscuola.it

IL VOLUME A COLPO D'OCCHIO



PER INTRODURRE ALL'UNITÀ

Contenuti digitali integrativi a disposizione dello studente

Prerequisiti

Test strutturato per la verifica dei prerequisiti

Conoscenze

Abilità

Competenze

PER APPRENDERE FACILMENTE

Esposizione chiara, con le definizioni evidenziate

Schemi e diagrammi per visualizzare i concetti

Per esempio, se Person è un'entità con caratteristiche nome e città, il cognome non è un'entità a sé stante.

Un'entità viene rappresentata graficamente con un rettangolo.

Attenzione! In molti casi si usa però il termine entità per indicare via via che cosa sia ormai comune, riducendo di generare quindi confusione; di solito se ne comprende comunque il significato dal contesto.

Attributi:

Gli attributi sono le proprietà attraverso cui è possibile descrivere un'entità (o un'unità).

Vengono rappresentati graficamente con dei punti collegati alle entità e alle associazioni. Non possono esistere due volti o sono identificati dal numero di attributo. I valori degli attributi sono sempre di tipo numerico, ma non necessariamente intero. La dimensione di un attributo è definita dall'insieme dei valori che possono essere assunti (per esempio, da "0" a "1" per l'attributo sesso).

A seconda del tipo di dato gli attributi possono essere di tre tipi:

- Attributi semplici:** sono quelli in cui esistono solo singoli componenti (per esempio una data, Fig. 1); in pratica si tratta di insiemni di attributi che possono essere considerati come un'unica proprietà dell'entità.
- Attributi composti:** sono quelli in cui esistono sottocomponenti (per esempio una data, Fig. 2); in pratica si tratta di insiemni di attributi che possono essere considerati come un'unica proprietà dell'entità.
- Attributi multipli:** sono quelli di cui nel titolo c'è già il suffisso (per esempio una lista, Fig. 3); in pratica si tratta di insiemni di attributi che possono essere considerati come un'unica proprietà dell'entità.

Le liste di attributi sono usate per descrivere un insieme di dati, come ad esempio la lista degli studenti di una classe o di una scuola. Per esempio, Studente, perché può essere suddiviso in giorno, mese e anno di nascita. Un altro esempio di attributo composto è indirizzo, che contiene al suo interno tra i numerosi dati.

Uno schema ER gli attributi complessi li rappresentiamo come nella Fig. 4.

Se necessario, il plug-in può essere scaricato dal server e automaticamente installato dopo il download. Il plug-in può operare all'interno di una pagina HTML, al fine di visualizzare dati in formato proprietario (per esempio, un plug-in AVANTIMPS consente di visualizzare filemif codificati in questo formato). Spesso però i filemif funziona solo su sistemi operativi specifici, utilizzando per la visualizzazione di un intero documento (per esempio Adobe Acrobat Viewer). Può anche essere eseguito in background (per esempio M3U player). Alcune applicazioni eseguono inoltre, la funzione di generazione di grafici (CAD), l'esecuzione di presentazioni interattive (phase), l'esecuzione di codice Java (applets).

Linguaggio Java:

Si tratta di un linguaggio inserito in pagine HTML. La funzione di questo linguaggio applicazioni consiste nell'introdurre estensioni all'interfaccia di navigazione web, cioè di inserire nuove funzioni nel browser.

È la esecuzione di una pagina web senza ricorrere allo sviluppo di plug-in o di applet Java, attività che richiede la competenza di un programmatore. Gli script possono essere eseguiti sia a livello server, sia a livello client. Quando si esegue un script a livello client non possono accedere alle risorse locali (sono quindi sicuri).

I principali linguaggi di scripting sono JavaScript e Visual Basic Script (VBScript).

JavaScript:

Dalla versione 11 di Internet Explorer, VBScript non viene più supportato. Il linguaggio JavaScript invece è molto più potente e più portatile. I vecchi script scritti in VBScript possono essere convertiti in JavaScript e viceversa.

Il linguaggio JavaScript è molto più potente e più portatile. I vecchi script scritti in VBScript possono essere convertiti in JavaScript e viceversa.

Il linguaggio JavaScript deriva dal Visual Basic. Come per JavaScript, il codice a inserire in pagine HTML, dopo il tag <SCRIPT LANGUAGE = "VBScript">, è eseguito nel browser. È un linguaggio molto simile a JavaScript, ma differisce da quest'ultimo nella portabilità (diametralmente inferiore a JavaScript, in quanto utilizzabile solo con Internet Explorer) e nella fase di apprendimento del linguaggio (diametralmente più facile di JavaScript).

Visual Basic Script (VBScript):

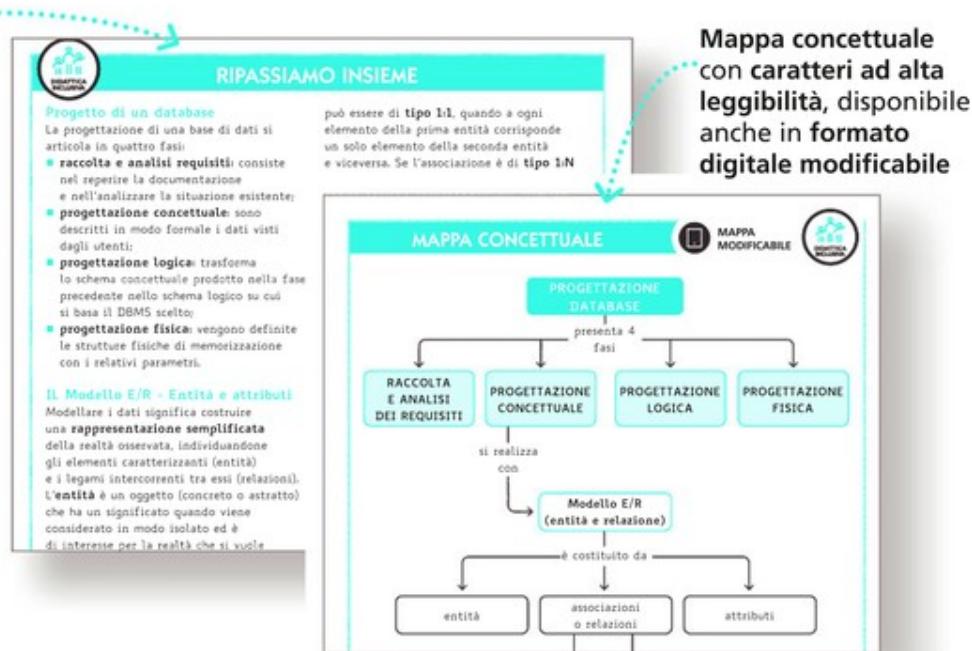
Il linguaggio VBScript deriva dal Visual Basic. Come per JavaScript, il codice a inserire in pagine HTML, dopo il tag <SCRIPT LANGUAGE = "VBScript">, è eseguito nel browser. È un linguaggio molto simile a JavaScript, ma differisce da quest'ultimo nella portabilità (diametralmente inferiore a JavaScript, in quanto utilizzabile solo con Internet Explorer) e nella fase di apprendimento del linguaggio (diametralmente più facile di JavaScript).

Domande di verifica immediata alla fine di ogni lezione

Box per evidenziare nozioni importanti

PER RIPASSARE IN OTTICA INCLUSIVA

Alla fine di ogni unità, sintesi di ogni lezione con caratteri ad alta leggibilità



PER VERIFICARE L'APPRENDIMENTO E SVILUPPARE LE COMPETENZE

Per ogni unità, un test strutturato per la verifica delle conoscenze, disponibile anche in versione digitale interattiva autocorrettiva

AUTOVALUTAZIONE DELLE CONOSCENZE

TEST

Vero o falso?

1. L'associazione ricorsiva è tra due entità.
2. Qualsiasi attributo di un'entità può essere di tipo 1:N.
3. Lo schema E/R è la rappresentazione concettuale di una base di dati.
4. Una gerarchia è una tipologia di associazione.
5. Un'associazione non può avere atti.

Scelta multipla (una sola è la risposta corretta)

6. Il modello E/R descrive:
 - lo schema fisico di una base di dati
 - una singola vista
 - lo schema concettuale di una base di dati
 - i dati di interesse
7. Gli attributi esprimono le caratteristiche:
 - delle chiavi primarie
 - delle relazioni
 - d'entità e relazioni
 - di entità e chiavi candidate
8. Un gerarchia ISA è:
 - completa e non esclusiva
 - esclusiva e non completa
 - non completa e non esclusiva
 - completa ed esclusiva

PREPARATI PER IL COLLOQUIO ORALE

1. Che cosa rappresenta il modello E/R [2]

Materiali per la didattica CLIL

Abstract dell'unità

File audio

Exercises

Glossario interattivo in italiano e in inglese con funzione di ricerca

Glossario su carta

GLOSSARIO CLIL

Entità any object or system that we hope to model. In general entities are irreducible concepts such as persons, place, things or relations which have relevance to the database.

Relazione represent a link between two or more entities.

Attributo a property inherent in a database entity or associated with that entity for database purposes.

Candidato chiave a set of data that can uniquely identify any database record without referring to any other data.

Chiave primaria a choice of candidate key; any other candidate key is called alternate key.

PER LE COMPETENZE TRASVERSALI E L'ORIENTAMENTO

Organizzazione dei percorsi

1. I PERCORSI PER LE COMPETENZE TRASVERSALI E PER L'ORIENTAMENTO

■ LE FINALITÀ DEI PERCORSI
Dal 2019 una legge approvata il 10 luglio 2019 n. 163 ha reso parte integrante del percorso scolastico nei diversi settori di istruzione e formazione professionale la dimensione delle competenze trasversali e l'orientamento. In questo caso si parla di "Percorsi per le competenze trasversali e per l'orientamento". In ogni caso si cerca, soprattutto nel mondo della formazione tecnica, di far acquisire ai giovani come le competenze trasversali e l'orientamento, sia attraverso la didattica tradizionale che attraverso la collaborazione e cooperazione con le aziende fra cui terzi di campo di studio, e per le quali è stata un'opportunità per comunicare e conoscere più da vicino il mondo del lavoro prima di concludere il percorso di formazione. Il percorso deve essere strutturato in modo tale da favorire la capacità di trovare nuove motivazioni allo studio e di cogliere come funziona il mondo del lavoro. In questi anni il percorso si completa e si perfeziona con nuove esperienze, che si svolgono, tra l'altro, in aziende e imprese, e si rinnova ogni anno con nuovi obiettivi e contenuti. Ricordiamo quanto ha fatto in questi anni e le modifiche con cui i percorsi riguardano le attività. Provate a considerare quanto ditemi con ciò che fate realmente sotto, in modo che sia in grado di riconoscere l'esperienza fatta e necessaria a un futuro collaudato di lavoro.

■ L'ORGANIZZAZIONE DEI PERCORSI
Le diverse modalità di organizzazione
In questi anni ha potuto evolversi le attività secondo vari modelli organizzativi. È stato infatti così, in collaborazione con aziende, che ha predisposto per le persone formative permanente, temporaneo o di inserimento, con specifiche modalità di organizzazione e di apprendimento, che sono serviti a soluzioni le sue attività e l'efficacia del processo formativo. Ha alternato periodi di attività a scuola con periodi in cui si è stato a diretto contatto con persone provenienti dal mondo del lavoro, che hanno fornito le loro conoscenze e le loro esperienze di lavoro. In questi anni si è anche lavorato di questo percorso di formazione le competenze che si acquisiscono sono state definite dal Consiglio di classe redigendo la certificazione delle competenze.

Tra le varie figure con cui può andare a fare nel corso di tali esperienze, due in particolare sono state attivate: il tutor sostanziale e quello di orientamento. In collaborazione con il Consiglio di classe, ha garantito la continuità dell'esperienza di lavoro e il rapporto con le aziende e i tutor. Il Project Work, mentre il Consiglio di classe ha seguito durante l'attività gli esercizi di controllo e di valutazione.

I due ruoli hanno proposto e verificato la sua attività di formazione fin dai primi momenti. In particolare, il tutor sostanziale li ha aiutati a valutare gli obiettivi raggiunti e le competenze che progressivamente erogavano, tenendo aggiornato costantemente il Consiglio di classe.

La relazione conclusiva
La relazione conclusiva è creata su misura del percorso scolastico per il conseguimento del diploma e della qualifica, sia per gli eventuali passaggi fra i sistemi, compresa la transizione nei percorsi di apprendimento. Al termine di ogni esperienza scolastica il tutor ha elaborato insieme un report sui tutti risultati il Consiglio di classe ha provveduto il compito, alla fine di ogni anno, di certificare le competenze da lui acquisite durante il percorso.

31a - Percorsi per le competenze trasversali e per l'orientamento

Valutazione delle competenze trasversali

2. RICONOSCERE E VALUTARE LE COMPETENZE TRASVERSALI

■ La logica principale dei percorsi per le competenze trasversali e l'orientamento è quella di permettere di rafforzare il processo di conoscenza e apprendimento che ha avuto luogo a scuola e di individuarne altre che si stanno coltivando nel mondo della formazione tecnica.
Questo è possibile attraverso la valutazione di tre grandi categorie, quelle relative ai conoscimenti e alle procedure disciplinari, che si deve sempre abituare a utilizzare a scuola (impostazioni didattiche) e quelle relative alle capacità relazionali, che poi molto meno adattano e utilizzano a scuola, ma che hanno un'importanza crescente nel mondo del lavoro (impostazioni trasversali).

Nel mondo della scuola sono privilegiate le competenze relative alla disciplina studiata. Nel mondo del lavoro sono richieste non solo queste competenze, ma anche molte altre. Tu hai contribuito a costruire conoscenze perché di queste competenze, ma anche molte altre, Tu hai contribuito a costruire anche altre. Questo è stato fatto attraverso la valutazione di queste competenze.

Quale sono tali competenze ulteriori? Come avrai avuto modo di sperimentare ci sono, per esempio, la capacità di lavorare in gruppo, di cooperare con gli altri sapendo fare capo a sé, la capacità di risolvere problemi complessi, la capacità di agire in modo autonomo e autonome, di impostare i tempi di svolgimento del lavoro, di affrontare gli imprevisti, di comunicare sui compiti che devono essere eseguiti.

■ IL SUPPORTO CON IL MONDO DEL LAVORO E LA DIDATTICA PER COMPETENZE
Oltre alle competenze disciplinari e scientifiche, di solito si aggiungono anche quelle necessarie per vivere in società, per poter fare un lavoro collettivo con gli altri, soprattutto: sapere accettare critiche, sapere rispettare i diritti di consenso, e, in generale, le competenze di cittadinanza (saper risolvere problemi, sapere avere un senso di responsabilità, migliorare se stesso).

Queste competenze sono importanti per il nostro lavoro quotidiano. Dicono anche di essere capaci di riflettere su se stessi, sulla sua capacità di essere autonomo e, di saperne conoscere se non compi, in genere di essere consapevole dei suoi punti di forza e di debolezza.

■ LA VALUTAZIONE DELLE COMPETENZE DI CITTAZIANZA
Data l'importanza che riveste la valutazione di cittadinanza mettiamo nella valutazione, ti proponiamo un esempio di modello per valutare:
La lettura dei livelli che puoi raggiungere chiaramente il significato delle competenze trasversali e il perimetro di provenire ad autocriticità.

31b - Percorsi per le competenze trasversali e per l'orientamento

Relazione conclusiva

3. ESEMPIO DI UNA RELAZIONE CONCLUSIVA SUL "PERCORSI"

Nel seguito mostriamo un esempio di relazione conclusiva sulle attività per le competenze trasversali e per l'orientamento. Per ogni parte della relazione verranno dati consigli su come realizzarla; è possibile a disposizione sul sito la relazione completa. L'esempio in basso ha una esperienza realmente effettuata da studenti che ne negli anni scorsi.

■ Introduzione, prima pagina e sommario

Potrai scrivere banali, ma che la prima pagina ben fatta permette già a chi legge la relazione di avere una visione generale di tutto. La prima pagina deve presentare tutte le informazioni necessarie, ma espresse in modo accattivante, per convincere il lettore ad aprire la relazione e a leggerla.

Dovrai indicare il nome, cognome, classe dello studente, nonché i riferimenti per identificare la scuola. Il titolo dovrà esser evidenziato e magari dovrà essere accompagnato da un simbolo appropriato.

RELAZIONE sui PERCORSI PER LE COMPETENZE TRASVERSALI E PER L'ORIENTAMENTO

Sommario:
1. Introduzione ...
2. Sommario ...
3. Contatti con le aziende ...
4. Conclusione ...
SOMMARIO:
1. Introduzione ...
2. Sommario ...
3. Contatti con le aziende ...
4. Conclusione ...
5. ...
6. ...
7. ...
8. ...
9. ...
10. ...
11. ...
12. Conclusioni personali ...
13. Autodenuncia ...



31c - Esempio di una relazione conclusiva sul "Percorsi" 301

PER LA PRESENTAZIONE DI UN PROGETTO

Progettazione della presentazione di un progetto

■ Basi teoriche

In questa parte andranno descritte tutte quelle informazioni di base necessarie per comprendere il lavoro presentato. Dovranno essere semplici, affinché le capisca anche chi non è un tecnico, ma precise e corrette, per mostrare le proprie competenze relative all'argomento in oggetto. Può essere utile dare alcune nozioni base iniziali per tutti e dedicare poi un sottoparagrafo più dettagliato per chi voglia approfondire. Meglio se poi il tutto viene corredata da immagini e schemi.

■ BASI TEORICHE

L'applicazione si compone di due parti: applicazione Android e OCR, che si basa pesantemente su tecniche di intelligenza artificiale, deep learning e reti neurali.

■ OCR

Un OCR (Optical Character Recognition) è un sistema che permette di convertire i caratteri contenuti all'interno di un'immagine in testo digitale comprensibile dal computer ed è caratterizzato dalle seguenti fasi:

1. rilevazione del testo (text detection);
2. segmentazione dei caratteri (charact segmentation);
3. classificazione dei caratteri (charact classification).



Fig. 1 Esempio di un OCR generico

■ INTELLIGENZA ARTIFICIALE, DEEP LEARNING E RETI NEURALI

L'OCR è realizzato utilizzando una rete neurale CNN che, dopo essere stata sviluppata, è esportata sul telefono. L'applicazione Android mostra in tempo reale la classificazione dell'immagine: infatti, per rendere più gradevole la user experience, non viene richiesto di scattare singole foto, ma il processo viene eseguito su un video in real time. Tutte le immagini vengono passate alla rete neurale e i risultati vengono stampati a video. L'output è formato da un valore di confidenza per ogni carattere, che è un numero da zero a uno, e vengono mostrati i caratteri con l'output più alto. Dato che non è implementata la rilevazione del testo e la segmentazione, per il corretto funzionamento dell'applicazione è necessario inquadrare un carattere per volta, affinché occupi l'intera schermata.

■ Sviluppo

Questa è la parte principale del lavoro. Qui andranno indicate le fasi di sviluppo del progetto, le informazioni tecniche, gli ambienti di sviluppo, e tutto quello che può servire a descrivere il lavoro. In questa parte è bene essere precisi e rigorosi dal punto di vista tecnico, anche se non sempre è necessario dilungarsi eccessivamente con sigle e termini.

Un esempio

2. REALIZZARE LA PRESENTAZIONE DI UN PROGETTO: UN ESEMPIO

■ Lo scopo della presentazione

Dopo aver stilato la relazione del progetto è consigliabile creare una presentazione con una decina di slide, che sarà di fondamentale aiuto durante l'esposizione orale. La presentazione serve:

- a te, per avere un riferimento per il tuo discorso, in modo da evitare scene mute; quindi la cosa più utile è inserire delle parole-chiave;
- agli ascoltatori, per seguire il filo del discorso; quindi deve essere accattivante ed esplicativa. Si può ottenere un effetto di questo tipo inserendo immagini (che spesso valgono più di mille parole).

■ Prima di iniziare

Il punto di partenza per la presentazione può essere la relazione, da cui devi scegliere le parti importanti, perché non avrai il tempo per raccontare tutto ciò che hai scritto. Una scelta sensata è quella di spiegare in modo particolarmente approfondito un pezzo del tuo progetto.

■ Il modello

I programmi di presentazione offrono svariati modelli: sceglie uno invitante, ma non complesso.

■ La sequenza delle diapositive

La presentazione non deve essere troppo lunga: dieci slide circa vanno bene. La struttura entro cui disporre in sequenza le diapositive è la seguente:

1. copertina;
2. introduzione;
3. svolgimento;
4. gestione del progetto;
5. conclusioni.

Ecco un esempio.

1. Copertina: dovrà contenere il titolo del progetto, eventualmente un sottotitolo, i propri nome e cognome.

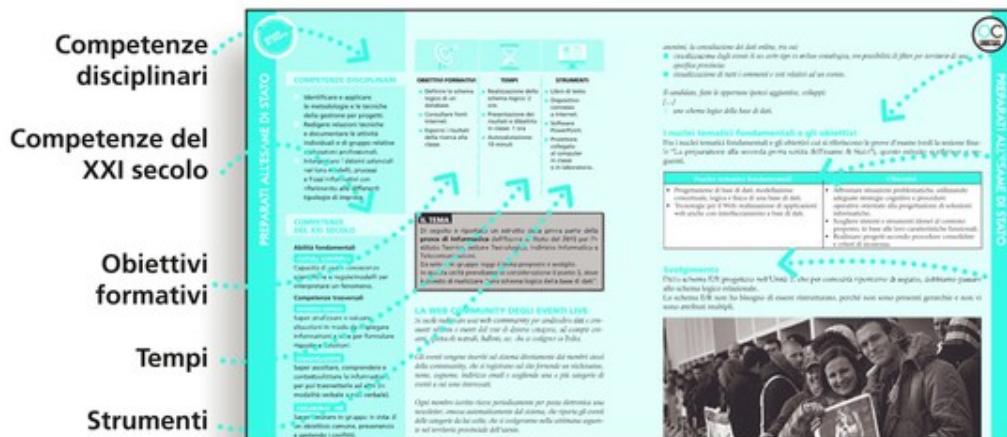
PROGETTO D'ESAME
Defranco Giada

Fig. 2 - Copertina

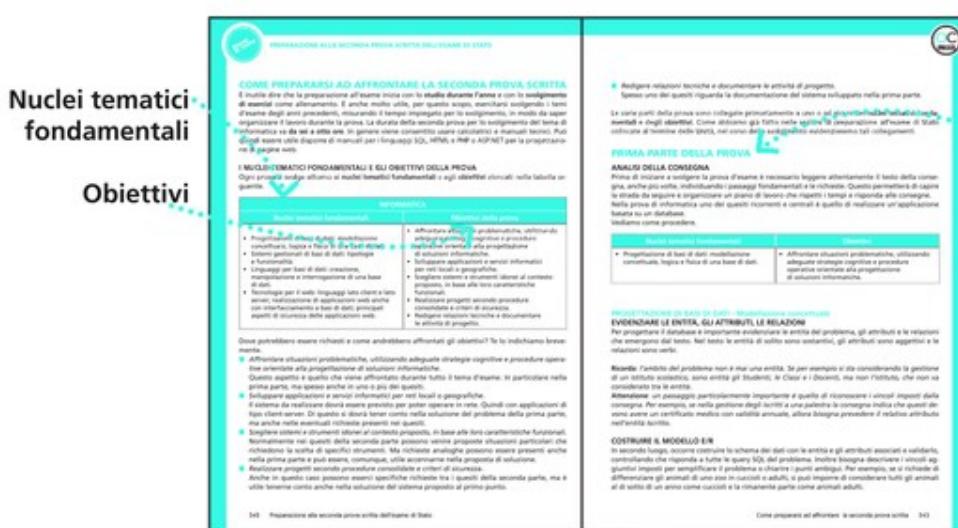
PER L'ESAME DI STATO

Alla fine delle unità

Schede per la preparazione della seconda prova scritta dell'esame di Stato



Alla fine del volume
Consigli generali per preparare la seconda prova scritta



1

Basi di dati



ESERCIZI COMMENTATI

Segui la risoluzione passo passo



PREREQUISITI

- Conoscere l'importanza dell'archiviazione dei dati.
- Conoscere il concetto di record e di campo.
- Saper gestire la memoria di massa.
- Saper definire la giusta struttura per rappresentare i dati.

PER COMINCIARE

1. Un tipo di memoria di massa è:

- | | |
|----------------------------------|------------------------------------|
| <input type="checkbox"/> A RAM | <input type="checkbox"/> C DVD |
| <input type="checkbox"/> B video | <input type="checkbox"/> D scanner |

2. Quali delle seguenti sequenze di memoria sono ordinate correttamente in ordine crescente di velocità?

- | |
|---|
| <input type="checkbox"/> A RAM, disco magnetico, disco ottico, memoria flash |
| <input type="checkbox"/> B Disco magnetico, disco ottico, RAM, memoria flash |
| <input type="checkbox"/> C Memoria flash, memoria cache, RAM, disco magnetico |
| <input type="checkbox"/> D Disco magnetico, disco ottico, memoria cache, RAM |

3. Definisci la struttura più adatta per rappresentare i professori della tua scuola.

4. L'organizzazione dei dati in memoria di massa viene gestita dal:

- | | |
|---|--|
| <input type="checkbox"/> A File system | <input type="checkbox"/> C Gestore periferiche |
| <input type="checkbox"/> B Schedulatore | <input type="checkbox"/> D Gestore CPU |

CONOSCENZE

- Memorizzazione dati nelle memorie di massa.
- Conoscere il concetto di database e di DBMS.
- Conoscere le funzionalità di un DBMS.
- Conoscere i concetti di base relativi ai principali modelli, linguaggi e sistemi per le basi di dati.
- Conoscere il modello ANSI/SPARC.
- Conoscere le principali problematiche relative alla sicurezza di un database.

ABILITÀ

- Saper riconoscere i vari modelli di un DBMS.

COMPETENZE

- Utilizzare le strategie del pensiero razionale negli aspetti dialettici e algoritmici, per affrontare situazioni problematiche, elaborando opportune soluzioni.
- Scegliere dispositivi e strumenti in base alle loro caratteristiche funzionali.
- Gestire progetti secondo le procedure e gli standard previsti dai sistemi aziendali di gestione della qualità.
- Gestire progetti secondo gli standard previsti dai sistemi aziendali di gestione della qualità e della sicurezza.

1. I DATI IN AZIENDA

■ L'azienda

In un'economia sempre più competitiva e in un mercato globale, il mantenimento e la distribuzione delle conoscenze aziendali (**know-how**) sono fattori chiave per avere successo. Il know-how è rappresentato generalmente da informazioni e documentazioni di natura eterogenea (dati, numeri, disegni, immagini, tabelle, norme e procedure), che non sempre sono facili da organizzare, archiviare e rendere disponibili.

Un'azienda che voglia raggiungere i propri fini deve dare particolare rilievo alle attività riguardanti la gestione delle informazioni. Tali attività hanno subito notevoli cambiamenti per quanto riguarda l'efficienza e la velocità di reperimento con l'introduzione della gestione automatica delle informazioni attraverso gli elaboratori.

In un primo tempo gli elaboratori sono stati usati soprattutto per gestire informazioni numeriche in attività standardizzate e ripetitive (calcolo degli stipendi, contabilità, ecc.); in seguito, con l'avvento dei personal computer e con la possibilità di collegare in rete gli elaboratori, si sono aperte possibilità di uso del calcolatore nei più svariati settori, ottenendo risultati neppure ipotizzabili negli anni precedenti. Ulteriori sviluppi dello scenario industriale, ma anche cambiamenti dei processi aziendali, sono stati indotti dall'avvento di Internet. In particolare sono nate opportunità nuove per lo sviluppo dei processi aziendali, come e-business ed e-commerce.

Lo scopo del lavoro finalizzato alla gestione delle **informazioni** è permettere a chi opera in azienda di prendere decisioni e fare le scelte più vantaggiose per l'azienda stessa. Poiché le scelte possono essere molto differenti tra loro, anche le informazioni prese in considerazione e le loro modalità di accesso devono variare.

Le **decisioni** che devono essere prese in azienda sono normalmente considerate di tre tipi: strategiche, tattiche e operative.

- Le decisioni **strategiche** definiscono le linee globali dell'azienda, sono generalmente prese a livello dirigenziale e necessitano di informazioni globali, ma sintetiche.
- Le decisioni **tattiche** sono prese e gestite dai quadri intermedi, che predispongono le strutture per il perseguimento delle decisioni strategiche, utilizzando informazioni a un livello intermedio.
- Le decisioni **operative**, prese dal personale esecutivo sulla base di informazioni analitiche, sono quelle che permettono di gestire le risorse per raggiungere gli obiettivi prefissati.

Dal punto di vista del personale e dei **ruoli**, l'azienda è in genere vista come una piramide alla cui base si possono collocare le persone con compiti esecutivi, mentre sono posti al vertice coloro che danno ordini (**fig. 1** alla pagina seguente).

LO SAI CHE

Se una volta il valore di un'azienda era dato solo dalle attrezzature e dal fatturato, adesso anche il know-how, cioè le conoscenze dell'impresa, concorrono a far crescere il suo valore.

Il flusso delle informazioni procede dall'alto verso il basso e viceversa, con un livello di dettaglio che varia a seconda della direzione (quando le informazioni salgono verso il vertice della piramide, aumenta la sintesi e diminuisce il dettaglio; quando scendono verso la base aumenta il dettaglio e diminuisce la sintesi) in analogia con le necessità delle figure professionali che vi operano. In particolare, le informazioni che vanno verso il basso accompagnano le decisioni e le indicazioni operative.

fig. 1 Flusso di informazioni e decisioni rispetto alla piramide aziendale nel modello Entità/Relazioni



Per capire meglio questo concetto, consideriamo l'esempio di un'azienda che vende prodotti di largo consumo distribuiti in tutta Italia.

Ogni rappresentante (livello esecutivo) ha a disposizione i dati dettagliati sulle vendite di ogni singolo prodotto e per ogni singolo cliente.

A livello intermedio i dati che interesseranno i responsabili di filiale sono raggruppati per prodotto rispetto a ogni rappresentante o a una particolare zona.

Questi dati vengono poi raggruppati e costituiscono i dati di vendita a livello regionale (sia divisi per prodotto sia globali).

I manager a livello nazionale vedranno solo questi ultimi dati e in base alle informazioni potranno prendere le decisioni. Per esempio considerando il calo delle vendite di alcuni prodotti in certe zone rispetto ad altre, potranno decidere di investire risorse finanziarie per spingere quei prodotti in quelle zone. La decisione/informazione sarà passata ai livelli intermedi, i quali, a partire dai dati più dettagliati a loro disposizione, decideranno come investire la somma prevista dal budget per spingere i prodotti (per esempio, con un ribasso del prezzo o incentivando i rappresentanti).

■ Il flusso informativo

Per produrre le informazioni, ogni azienda ha la necessità di raccogliere e organizzare i dati, cioè elementi grezzi derivati da determinati fenomeni. Per esempio, la quantità di beni venduti dall'azienda in un determinato periodo è un dato.

I dati costituiscono una risorsa per l'azienda, un patrimonio significativo da sfruttare e proteggere.

I dati sono più stabili delle applicazioni che operano su di essi.

Per esempio, i dati relativi alle applicazioni bancarie hanno una struttura sostanzialmente invariata da decenni, mentre le procedure che agiscono su di essi variano frequentemente.

I dati possono provenire dall'ambiente interno, e rappresentano fenomeni aziendali quantitativi e qualitativi, o dall'ambiente esterno, e rappresentano fenomeni e situazioni esterne all'azienda.

Ricordiamo che i dati non sono informazioni, ma permettono di creare informazioni.

Per diventare informazioni, devono essere rielaborati, combinati tra loro e interpretati in modo da produrre le conoscenze necessarie per le decisioni da prendere.

Per esempio, il confronto in un certo periodo tra le vendite dell'impresa e le vendite delle imprese concorrenti produce un'informazione che può essere utilizzata dai vertici aziendali per le decisioni strategiche.

In azienda il flusso delle informazioni deve seguire determinate regole. I dati devono giungere dove sono necessari, senza ritardi, in modo semplice e comprensibile, e soprattutto in modo affidabile.

Allo stesso modo gli utenti devono poter raggiungere agevolmente le informazioni di cui hanno necessità, scegliendo tra quelle messe a disposizione.

LO SAI CHE

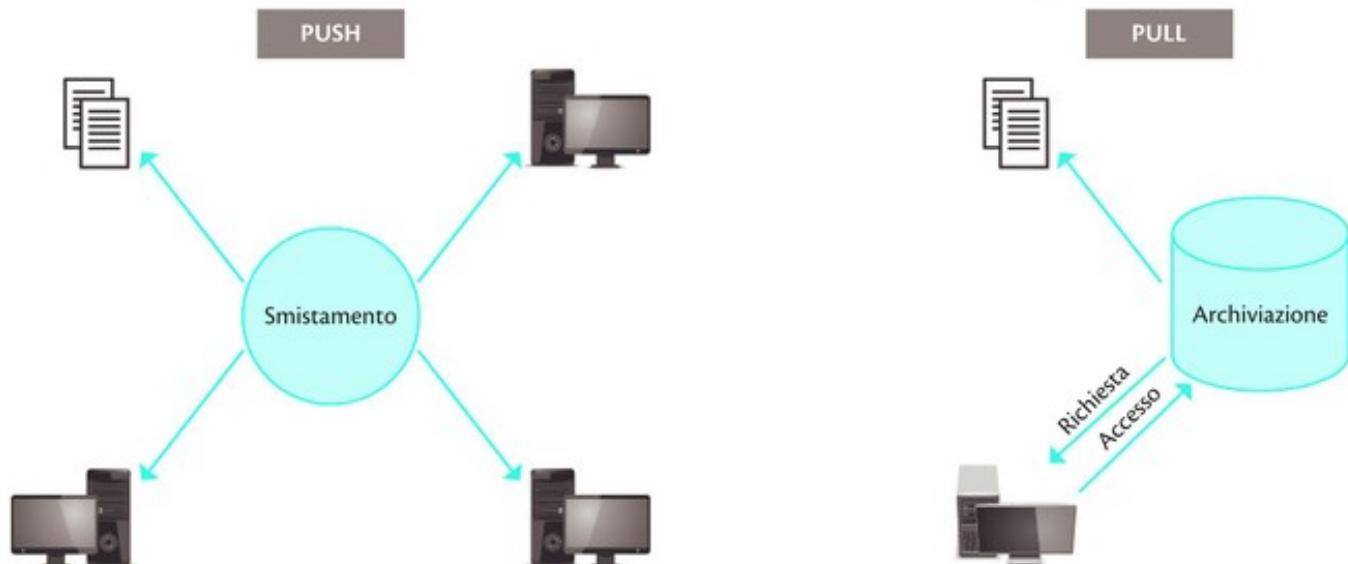
Un'informazione è un insieme di dati interpretati, contestualizzati, rielaborati e combinati tra loro.

Esistono quindi due casi (fig. 2):

- l'informazione, quando è disponibile, viene inviata a chi ne ha bisogno, subito o in un immediato futuro (push);
- l'informazione viene messa a disposizione in luoghi prefissati e sarà l'utente ad accedervi quando ne avrà bisogno (pull).

Facciamo un paragone con le riviste che arrivano in azienda: le pubblicazioni specialistiche sono consegnate alle persone che si occupano di quell'argomento, mentre quelle di carattere generale sono collocate in biblioteca a disposizione di tutti.

fig. 2 I flussi informativi



■ Il sistema informativo aziendale

Il sistema informativo

Per agevolare le attività (sia a livello individuale sia in organizzazioni di dimensioni piccole o grandi) è essenziale, oltre alla disponibilità delle informazioni, la capacità di gestirle in modo efficace. Ogni azienda deve essere dotata quindi di un **sistema informativo**, che organizza e gestisce le informazioni necessarie per perseguire gli scopi preposti. Il sistema informativo aziendale è l'insieme coordinato delle risorse umane, degli strumenti e delle procedure il cui compito è raccogliere, memorizzare ed elaborare i dati per produrre informazioni per i soggetti, interni ed esterni, interessati alla vita dell'impresa (*stakeholders*).

Lo scopo del sistema informativo è quello di produrre le informazioni di supporto alle decisioni aziendali, favorire la comunicazione delle informazioni all'interno dell'azienda e con i soggetti esterni e svolgere analisi sull'andamento dell'impresa.

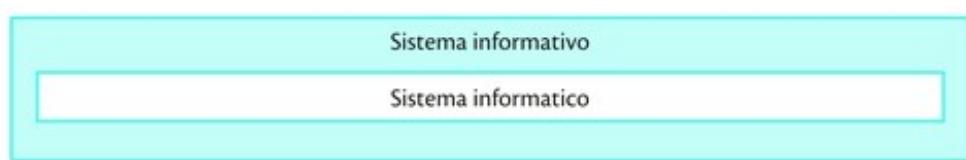
Il sistema informativo automatizzato

Si dice **sistema informatico** o **sistema informativo automatizzato** l'insieme delle tecnologie hardware e software che permettono di processare i dati aziendali e di automatizzare le funzioni del sistema informativo aziendale (fig. 3).

Il sistema informatico permette la produzione e la gestione di notevoli quantità di dati, riduce gli errori umani, elabora e smista le informazioni con tempestività ovunque siano richieste.

Per esempio, se consideriamo la quantità di dati da gestire per la contabilità aziendale, notiamo quanto sia importante l'utilizzo di procedure informatiche automatizzate per produrre i bilanci e la gestione di ordini, fatture, acquisti e buste paga, con notevole risparmio di tempo e costi per l'azienda rispetto ai metodi tradizionali cartacei.

fig. 3 Sistema informatico e sistema informativo



Gli strumenti dell'ICT

Affinché un'azienda abbia successo, è importante che manager e tecnici siano in grado di sapere in ogni momento come si sta muovendo l'azienda stessa, come reagiscono i clienti, come si opera con i fornitori.

Queste informazioni sono ancora più importanti per i vertici aziendali, dove sono prese le decisioni strategiche.

Così, per facilitare la comunicazione tra i ruoli e il coordinamento dei processi aziendali esistono appositi strumenti informatici (i cosiddetti **strumenti dell'ICT**), schematizzati nella fig. 4 e descritti alla pagina seguente.

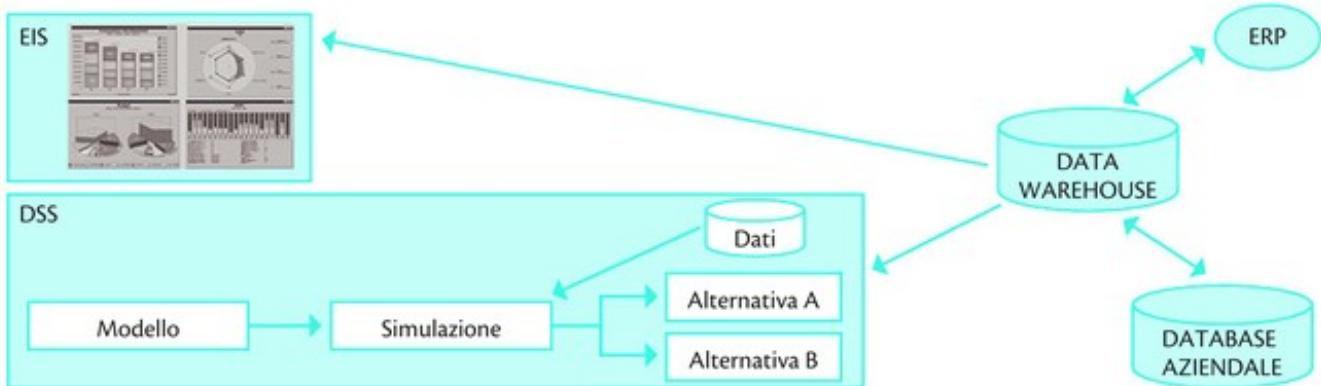


fig. 4 Data warehouse, EIS, DSS ed ERP

DATA WAREHOUSE

Un *data warehouse* è progettato per avere a disposizione informazioni di sintesi e di dettaglio estratte dalle diverse basi di dati aziendali (o esterne) e organizzate in modo da facilitarne la ricerca e l'inclusione in modelli decisionali (DSS), in «cruscotti», tabelle, grafici, report di controllo (EIS) e per le applicazioni di e-business.

DATA MINING

Spesso, a fronte di risultati al di fuori dei limiti di normalità, si applicano tecniche e strumenti matematici e statistici molto sofisticati (i *data mining*) per permettere l'interpretazione dei comportamenti aziendali e aiutare a prendere decisioni più corrette.

DSS

I DSS (*Decision Support System*) sono applicazioni costituite da un'interfaccia utente adatta a gestire un insieme di modelli e metodi e opportune basi di dati (data warehouse) che aiutano a valutare alternative decisionali tramite un'approfondita analisi dei dati storici, elaborati mediante modelli d'analisi e predittivi per il futuro.

EIS

Un EIS (*Executive Information System*) fornisce l'accesso a informazioni rilevanti e utili secondo modalità semplici e tempestive di navigazione. Attraverso l'elaborazione di dati elementari, permette di creare indicatori e report di controllo direzionale e operativo.

ERP

Gli ERP (*Enterprise Resource Planning*) sono sistemi amministrativi integrati che presentano in un unico software le attività di gestione e l'automazione delle operazioni gestionali, in modo da garantire un maggior controllo di gestione.

FISSA LE CONOSCENZE

1. Che cosa significa ICT?
2. Che cosa si intende per e-business?

2. MEMORIZZARE DATI

■ I file

La conservazione di grandi volumi di dati e informazioni avviene attraverso l'uso di archivi, i **file**, registrati sulla memoria di massa per consentirne la conservazione per un periodo di tempo indeterminato.

Esistono diverse tipologie di file:

- **file di testo**, cioè sequenze di caratteri in formato ASCII, in genere organizzate per righe e spesso utilizzate per memorizzare i dati di input e di output dei programmi;
- **file binari** (o non strutturati);
- **file strutturati**, cioè insiemi di record memorizzati in codifica binaria. Un **record** è un insieme di informazioni (attributi) riguardanti un determinato oggetto. Le informazioni che compongono il **record** sono dette **campi del record** (nome, cognome, ecc.).

■ File strutturati

Un **archivio o file** (termine inglese per identificare un archivio) è definito come un insieme di record memorizzati su un supporto di memoria permanente. Per **record** si intende un insieme di informazioni (attributi) riguardanti un determinato oggetto.

Una persona per esempio è un "oggetto" che può essere caratterizzato da:

- nome;
- cognome;
- indirizzo;
- città;
- telefono;
- codice fiscale;
- data di nascita.

Le informazioni componenti il record sono dette **campi del record** (nome, cognome ecc.).

Il campo di un record può a sua volta essere suddiviso in sottocampi; per esempio, il campo data di nascita può essere rappresentato attraverso il giorno, il mese e l'anno.

Per maggiore chiarezza diremo che un file contiene sempre dati della stessa natura (omogenei), di cui si registrano le caratteristiche (attributi) omogenee, scelte in vista dell'obiettivo da raggiungere. Nell'esempio visto precedentemente non abbiamo considerato tutte le caratteristiche della persona, ma solo quelle che possono essere utili in funzione di quello che dobbiamo realizzare. Per definire le caratteristiche dei dati da rappresentare è necessario avere ben chiaro l'uso che si deve fare dei dati rappresentati. Per esempio, per un medico l'archivio persone dovrà

contenere tutti i dati relativi alla salute delle persone stesse; per l'ufficio erariale, invece, l'archivio persone dovrà contenere tutti i dati relativi alle loro entrate, alle loro proprietà e così via.

In pratica nella progettazione di un archivio è necessario definire quali sono gli attributi generali (campi) che meglio rappresentano i dati in oggetto. È necessario definire la struttura del record e cioè quali sono i campi e gli eventuali sottocampi che lo compongono, il loro tipo e la loro dimensione in byte. La dimensione dipende dalla lunghezza della stringa, se si tratta di un campo alfanumerico, e dal numero massimo rappresentabile se si tratta di un numero.

L'esempio della [fig. 5](#) mostra il tracciato per il record *dipendente* che contiene i campi:

[fig. 5](#) Struttura del record *dipendente*

DIPENDENTE

Codice	Cognome	Nome	DATA-ASS			Livello
			AA	MM	GG	
5 byte	20 byte	20 byte	4 byte	2 byte	2 byte	1 byte
CODICE (<i>codice del dipendente</i>)		alfanumerico	5 byte			
COGNOME (<i>cognome del dipendente</i>)		alfanumerico	20 byte			
NOME (<i>nome del dipendente</i>)		alfanumerico	20 byte			
DATA-ASS (<i>data di assunzione del dipendente</i>)	suddiviso in:					
	AA (<i>anno di assunzione del dipendente</i>)	numerico	4 byte			
	MM (<i>mese di assunzione del dipendente</i>)	numerico	2 byte			
	GG (<i>giorno di assunzione del dipendente</i>)	numerico	2 byte			
LIVELLO (<i>livello contrattuale del dipendente</i>)		numerico	1 byte			

Per migliorare l'attività di ricerca può essere utile, nella definizione di un file, definire una **chiave primaria**, cioè uno o più campi che identificano univocamente ciascun record; i valori assunti dalla chiave primaria devono essere tutti diversi: non possono esistere due record con lo stesso valore della chiave. Nell'esempio precedente il campo *codice* può essere assunto come chiave primaria, perché non esistono due persone con lo stesso codice, mentre il campo *cognome* non può essere chiave primaria, dato che possono esistere casi di omonimia.

■ Operazioni fisiche

Prima di parlare delle operazioni che si possono effettuare su un file è necessario ricordare che i file sono memorizzati in una memoria di massa, cioè in una memoria aggiunta al calcolatore. Per poter elaborare tramite un programma i dati presenti nei file è necessario prima spostarli nella memoria centrale.

È compito del sistema operativo (*file system*) trasportare fisicamente i dati dalla memoria di massa alla memoria centrale e viceversa. Queste funzioni vengono richiamate con i comandi READ e WRITE previsti in tutti i linguaggi di programmazione, anche se con differenti sintassi.

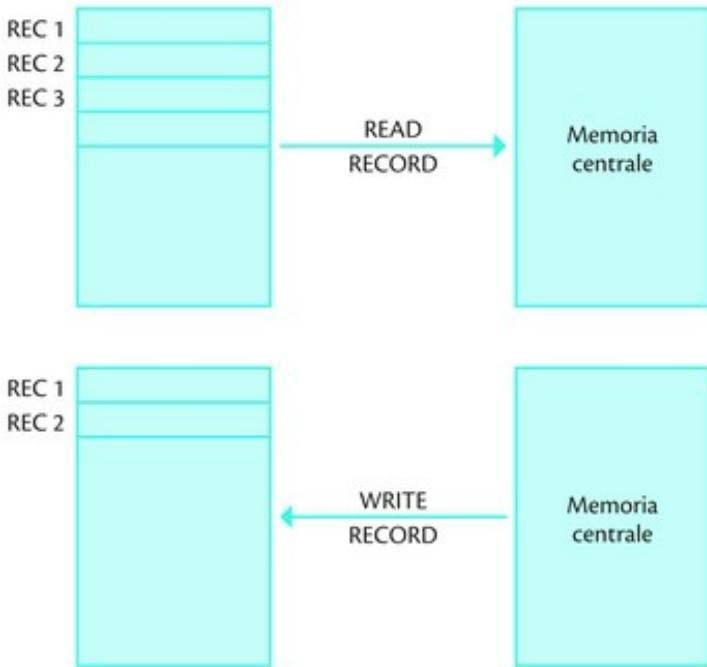


fig. 6 Trasferimento dei dati con le operazioni di *read* e *write*

In particolare con un'operazione di **READ** viene fatto un accesso alla memoria di massa e viene ricopiatò un record dalla memoria di massa alla memoria centrale. Con l'operazione di **WRITE** avviene il contrario e cioè il record presente nella memoria centrale viene copiatò nella memoria di massa (**fig. 6**). Finora abbiamo parlato di record come insieme delle informazioni relative a un'entità logica, definita a seconda delle esigenze dell'applicazione. Quella che è stata data fino a ora è in realtà la definizione di **record logico**, cioè della descrizione di come il programmatore, e quindi l'utente del programma, vede suddiviso l'insieme dei caratteri del file. Il record logico ha una lunghezza di byte pari alla somma della dimensione dei vari campi e tale lunghezza è stabilita dall'utente stesso.

Quello che invece viene trasferito in realtà è il **record (o blocco) fisico**, che indica la quantità di byte, di lunghezza prefissata, che è trasferita dalla memoria centrale alla memoria di massa o viceversa con un'unica operazione di I/O. La dimensione del record fisico dipende dalla struttura hardware della macchina.

A seconda dell'uso che se ne deve fare (gestione da parte del *file system* o uso in un programma), il file può essere considerato diviso in record logici o record fisici, e vi si possono fare differenti operazioni.

La **fig. 7** mostra come lo stesso file viene visto dal programmatore e dal *file system*. Il primo vedrà un insieme di record composti da due campi (nome di 7 caratteri ed età di 2), mentre il *file system* vedrà dei blocchi (record fisici) di 18 byte.

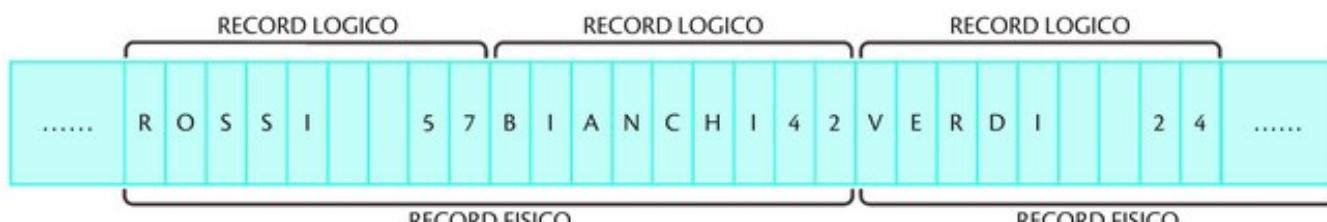


fig. 7 Record logico e record fisico

Per fattore di blocco si intende il numero di record logici contenuti in un record fisico. A seconda del fattore di blocco i record di un file si dicono:

- sbloccati, se ogni record fisico contiene un solo record logico (fattore di blocco = 1);
- bloccati, se ogni record fisico contiene più di un record logico (fattore di blocco > 1);
- multiblocco, se occorrono più record fisici per memorizzare un record logico (fattore di blocco < 1).

Poiché l'accesso alla memoria di massa ha tempi decisamente più elevati rispetto all'accesso alla memoria centrale o ai tempi di elaborazione della CPU, si cerca di minimizzare il più possibile i primi avendo dei record fisici abbastanza grandi. Le operazioni di lettura e scrittura interessano il record fisico, che è parcheggiato in un'area di memoria centrale (buffer). Se il fattore di blocco è 3, vuol dire che 3 record logici vengono copiati dalla memoria di massa nel buffer.

L'istruzione di READ comporta il trasferimento dalla memoria di massa alla memoria centrale solo quando il record logico richiesto non è presente nel buffer (fig. 8). Se invece il record è presente nel buffer, l'istruzione di READ comporta solo un trasferimento nella memoria centrale: dal buffer all'area di lavoro assegnata al programma.

Anche l'istruzione di WRITE comporta il trasferimento dei dati dalla memoria centrale alla memoria di massa solo quando il buffer è pieno. In caso contrario si tratta nuovamente di trasferimento in memoria centrale: dall'area di lavoro assegnata, al programma, al buffer.

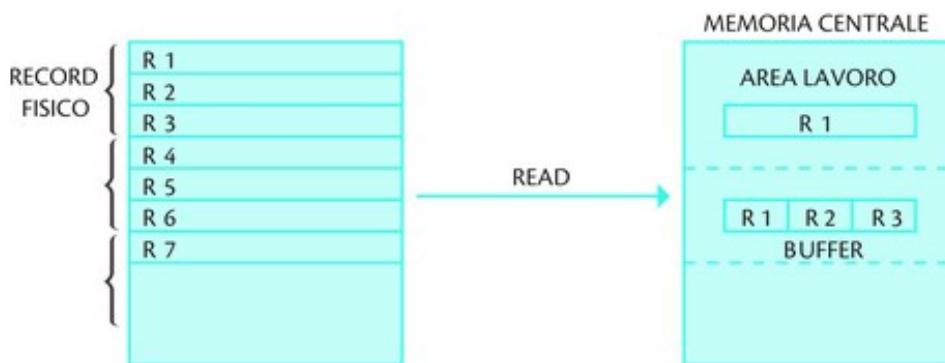


fig. 8 Operazione di lettura

■ Organizzazione dei file

Il metodo di organizzazione di un file stabilisce la logica e le modalità con cui le informazioni sono memorizzate sui supporti fisici.

I tipi di organizzazione possibili sono:

- sequenziale;
- ad accesso diretto.

Tra le modalità di accesso diretto vi è:

- accesso relativo (*relative*);
- accesso su indice (*indexed*);
- accesso calcolato (*hash*).

■ L'organizzazione sequenziale

Con il termine di file **sequenziale** si intende un tipo di archivio in cui i dati sono disposti in sequenza, cioè uno di seguito all'altro. Un file organizzato in modo sequenziale (in inglese *flat file*, cioè file piatto) consiste in un insieme di record contigui.

Si parla di file **ordinato** quando i record sono registrati in ordine crescente (o decrescente) rispetto al valore di un campo chiave; quando invece i record sono memorizzati consecutivamente, senza nessun ordine logico, il file si dirà **non ordinato**.

In un file sequenziale l'accesso a una data informazione (record) può avvenire solo scorrendo tutti i record che precedono quello ricercato e l'inserimento di un nuovo record può avvenire solo al fondo del file.

LO SAI CHE

Nel caso della ricerca di record con una data caratteristica il costo di una ricerca è uguale al numero di record che compongono il file (N). La ricerca termina quando non ci sono più blocchi. La ricerca di un record con un dato valore per la chiave richiede in media $(N + 1)/2$ accessi e quindi il costo della ricerca va da un minimo di 1 accesso alla memoria di massa (quando il record è il primo) a un massimo di N accessi (quando il record è l'ultimo).

fig. 9 Cancellazione logica

■ La ricerca nei file sequenziali

Se il campo su cui deve essere effettuata un'operazione di ricerca non identifica univocamente un record nel file, occorre scandire tutto il file sequenzialmente.

Se invece la chiave di ricerca identifica univocamente un record nel file, si può interrompere la ricerca quando è stato trovato il record o quando la chiave considerata è maggiore di quella da ricercare.

■ Cancellare un record

Per quanto riguarda la cancellazione i record possono essere cancellati solo logicamente e non fisicamente. Questo avviene inserendo nel tracciato un campo (flag_cancellazione) che indica se il record è stato cancellato (1) oppure no (0).

Nella fig. 9 è stato aggiunto il flag di cancellazione al file di fig. 5; i record con chiave 144 e 133 sono stati logicamente cancellati.

Quando si scorrerà il file per esaminare i record, quelli marcati con il flag verranno saltati in fase di lettura.

100 ...	0	105 ...	0	128 ...	0	110 ...	0
123 ...	0	144 ...	1	133 ...	1	120 ...	0
139 ...	0	131 ...	0	146 ...	0		
	1		2		3		4

■ Inserire un record

L'inserimento di un record avviene normalmente inserendo il nuovo record alla fine del file (metodo dell'*ampliamento*).

È possibile anche recuperare spazio inserendo il nuovo record nel primo record che risulta logicamente cancellato e sostituendo questo record con quello da inserire (metodo della *sovraposizione*). Questo può avvenire in modo semplice se il file non è ordinato.

FISSA LE CONOSCENZE

1. Che cosa si intende per archivio di dati strutturato?
2. Che cosa si intende per chiave primaria?
3. Quali operazioni fisiche sono possibili sui file?
4. Che cosa si intende per record logico? E per record fisico?
5. Che cosa si intende per file sequenziale?
6. Come avviene l'inserimento di un record in un file sequenziale?

3. I FILE AD ACCESSO DIRETTO

■ Accesso su indice

Per ovviare ai problemi legati ai file ad accesso sequenziale, è possibile provvedere alla indicizzazione dei record e accedere così direttamente a un record del file tramite il valore del campo *chiave*.

Una chiave è un campo che individua univocamente un record; un record può assumere un solo valore per il campo chiave e, all'interno del file, non possono esserci due record che presentano lo stesso valore. In questo tipo di organizzazione le chiavi sono memorizzate in un file detto *file degli indici*, per mezzo del quale è possibile ottenere informazioni utili per accedere al record desiderato, che si trova invece nel *file dei dati* (fig. 10). Il file degli indici è memorizzato in memoria di massa ed è ordinato per chiavi crescenti; può essere copiato automaticamente in memoria centrale in una tabella (da qui il nome di *tabella degli indici*) nel momento in cui si richiede l'uso dei dati del file.

Poiché la lettura in memoria di massa è molto più lenta rispetto a quella in memoria centrale, memorizzando il file indice in memoria centrale, si riducono i tempi di ricerca. Quando si deve cercare un record, si ricerca la chiave nella tabella degli indici in memoria centrale e solo quando la chiave è stata trovata si accede al blocco corrispondente in memoria di massa (il numero del blocco è associato alla chiave nella tabella degli indici) e si può quindi accedere al record completo nel file. In questo modo si effettua un unico accesso alla memoria di massa, riducendo i tempi di ricerca. La tecnica di spostamento della tabella degli indici in memoria centrale è applicabile solo se il numero dei record non è eccessivo, altrimenti si occuperebbe troppo spazio per la memorizzazione.

A una maggiore velocità di ricerca si contrappone un maggior consumo di memoria, nonché una maggiore complessità dei programmi che operano sui dati.

La fig. 10 presenta la situazione in cui i record con chiave A3, B7, C4 sono memorizzati nel blocco 1 mentre i record con chiave C8, D4, D6 sono memorizzati nel blocco 2. Quando in un file degli indici sono riportate le chiavi di tutti i record del file dei dati (come in questo caso), si parla di *indice denso* (fig. 10).

CHIAVE	N. BLOCCO
A3	1
B7	1
C4	1
C8	2
D4	2
D6	2

File degli indici

blocco 1	CHIAVE	DATI	
	B7	2.500	7
	C4	3.000	22
	A3	7.800	36
blocco 2	D6	240	5
	C8	364	8
	D4	2.530	16

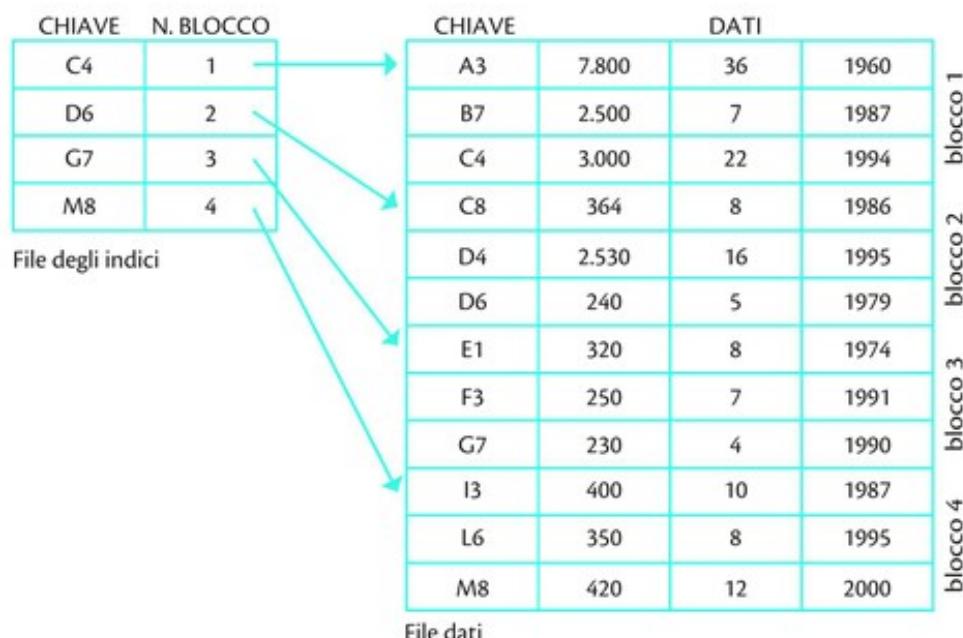
File dati

fig. 10 Indice denso

Quando il file dei dati è ordinato per chiave (fig. 11) e il numero di record

è notevole, allora si può utilizzare un *indice sparso*. Cioè si crea un file degli indici, in cui sono riportate solo le chiavi maggiori contenute in ciascun blocco del file di dati.

fig. 11 Indice sparso



LO SAI CHE

Dopo un certo numero di inserimenti nelle aree di *overflow* il sistema perde di efficienza, sia perché le aree di *overflow* possono essere piene, sia perché il reperimento di un record può costare numerosi accessi alla memoria di massa per seguire le catene dei puntatori.

Inserimento di un record

L'operazione di inserimento all'interno di un file sequenziale a indici può richiedere tempi di esecuzione onerosi, in quanto deve essere mantenuto l'ordinamento dei record anche dopo l'inserimento. Vedremo nel seguito alcune delle tecniche più diffuse per l'inserimento di elementi in un file ordinato: *overflow* e spazio distribuito.

Area di overflow

È possibile riservare un'area sulla memoria di massa che viene utilizzata per memorizzare i nuovi record da inserire nel file. In quest'area (*area di overflow*), che è inizialmente vuota, l'ordinamento viene mantenuto attraverso puntatori che permettono il collegamento a catena tra gli elementi.

Se l'inserimento dei record con chiave A4, D8 e A5 nel file di [fig. 11](#) avviene utilizzando l'area di *overflow*, la situazione dopo l'inserimento è illustrata in [fig. 12](#). L'inserimento della chiave D8 comporta l'aggiornamento del file degli indici.

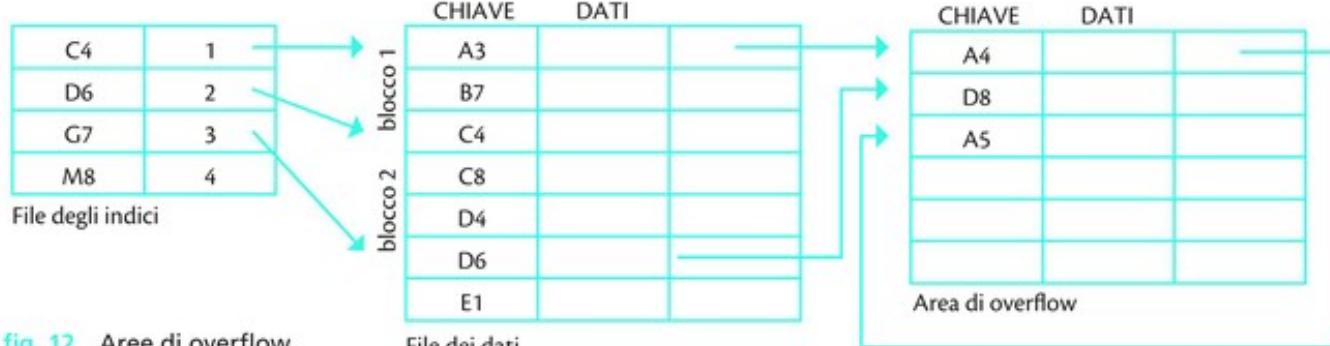


fig. 12 Aree di overflow

Spazio libero distribuito

Con questa tecnica alla fine di ogni blocco vengono riservati record vuoti, che saranno occupati dai nuovi inserimenti. Nel momento in cui si vuole inserire un nuovo elemento, si cerca il blocco che dovrà contenerlo e lo si inserisce nella posizione corretta, spostando eventualmente alcuni record del blocco in questione per mantenere l'ordinamento delle chiavi all'interno del blocco. Quest'ultima operazione è poco costosa, perché il blocco si trova già in memoria centrale e quindi non sono richiesti ulteriori accessi alla memoria di massa.

LO SAI CHE

Usando la tecnica dello spazio libero distribuito risulta più veloce l'operazione di ricerca, in quanto non è necessario scandire la catena di puntatori per trovare il record; ma presenta l'inconveniente di occupare più spazio di memoria.

■ Accesso calcolato

Un file si definisce ad accesso calcolato quando è possibile accedere direttamente alle informazioni contenute in un record attraverso la sua chiave. Il modo più rapido per individuare l'indirizzo del record voluto sarebbe quello di usare l'indirizzo fisico stesso come valore della chiave. Ovviamente ciò non è ragionevole, perché la chiave deve mantenere un valore logico per il livello applicativo. Sono state pertanto studiate funzioni matematiche apposite, dette funzioni di accesso o *funzioni hash*, che permettono di trasformare la chiave di un record nell'indirizzo di un'area (*bucket*) della memoria di massa in cui memorizzare (e successivamente reperire) il record stesso (fig. 13). In tal modo è sufficiente usare la stessa funzione hash sia per memorizzare sia per reperire le informazioni.



fig. 13 Trasformazione della chiave

Un file *hash* è suddiviso in aree, chiamate *bucket*, ciascuna delle quali è costituita da uno o più blocchi fisici che contengono i record logici (fig. 14). Il numero di record logici presenti in un *bucket* costituisce la capacità del *bucket*. Se N è il numero dei *bucket*, questi vengono numerati da 0 a $N - 1$.

fig. 14 Organizzazione di un file hash



■ Gestione dell'overflow

Nell'applicazione di una funzione *hash* può capitare di ottenere lo stesso indirizzo partendo da due chiavi diverse (*collisione*). Due o più chiavi per le quali la funzione *hash* genera collisioni si dicono *sinonimi*. Il fenomeno delle collisioni genera problemi di memorizzazione quando il *bucket* è pieno e per risolverlo si è costretti a ricorrere a tecniche chiamate *tecniche di gestione dell'overflow*.

Overflow aperto

In caso di collisione il nuovo record è aggiunto nel primo *bucket* non pieno, successivo al *bucket* pieno. È necessaria una scansione dei *bucket*,

che termina quando ne viene individuato uno non pieno. Con questo metodo non è necessaria l'area di *overflow*, perché anche i record con valori sinonimi delle chiavi sono memorizzati nell'area primaria.

Catene

Con il metodo delle catene viene associata al file *hash* un'area di *overflow*: a ogni *bucket* dell'area primaria è associato (tramite un puntatore) un *bucket* nell'area di *overflow* che può contenere solo i sinonimi dell'area primaria a esso associata. Quando anche il *bucket* di *overflow* è riempito, viene individuato un altro *bucket* nell'area di *overflow* che conterrà ancora record sinonimi. Tramite puntatori è possibile passare da un *bucket* all'altro.

Qualsiasi metodo si utilizzi per il trattamento delle collisioni, il risultato finale sarà che l'accesso ai dati, in presenza di collisioni, non sarà più un accesso diretto nel vero senso del termine, ma comporterà la scansione di più *bucket*. L'efficacia della tecnica di *hash* adottata dipende anche dalla capacità del *bucket* (che in genere coincide con un blocco fisico o una traccia): una capacità troppo bassa porterebbe a un elevato costo per gestire le collisioni, d'altra parte, una capacità troppo elevata porterebbe a un aumento eccessivo del tempo necessario per ricercare un record all'interno del *bucket*.

■ File relative

Con questo tipo di organizzazione è possibile accedere direttamente al record conoscendo la posizione relativa (1° record, 2° record ecc.) che occupa all'interno del file: questo tipo di organizzazione è possibile solo quando la chiave corrisponde alla posizione del record all'interno del file. Un esempio di file *relative* può essere quello contenente le informazioni relative ai posti di un aereo. I posti su un aereo sono numerati da 1 a N: ciascun record rappresenta un posto, che può essere quindi individuato tramite il suo numero.

Poiché tutti i record devono avere la stessa lunghezza (LR), è possibile conoscere la posizione che il record di chiave N occupa all'interno del file applicando la formula:

$$\text{POSIZIONE} = (N - 1) * \text{LR}$$

L'organizzazione *relative* permette sia l'accesso diretto sia quello sequenziale (in ordine di numero di posizione).

FISSA LE CONOSCENZE

1. A che cosa servono le funzioni di *hash* e che cosa sono i sinonimi?
2. Come si accede a un record memorizzato in un file *relative*?
3. Che cosa si intende per accesso diretto?
4. Che cosa contiene il file degli indici?
5. Come si realizza l'inserimento di un nuovo record in un file *hash*?

4. DAL FILE SYSTEM ALLE BASI DI DATI

■ Il sistema EDP

Per capire i motivi per cui si è arrivati alla necessità di creare le basi di dati, vediamo come si opera in un'azienda in cui il sistema informativo viene gestito utilizzando strumenti informatici. Questo sistema prende il nome di **EDP** (Electronic Data Processing) e può essere visto come un insieme di *dati* memorizzati su supporti elettronici e di *applicazioni informatiche* utilizzate per agevolare il raggiungimento degli obiettivi di un'azienda.

Ricorda che per **applicazione informatica** si intende una componente del sistema informativo (SI), che utilizza dati in esso immagazzinati per compiere una funzione specifica all'interno dell'organizzazione a cui il SI appartiene (per esempio la stampa di statistiche).

Nel sistema EDP ogni singola applicazione del SI opera su un insieme di dati memorizzati secondo una struttura definita all'interno dell'applicazione stessa dall'analista, il quale ha il compito di realizzare l'applicazione in modo da rendere massima l'efficienza della stessa, indipendentemente dalle altre applicazioni (massima velocità, minima occupazione di memoria e così via). Lo strumento utilizzato per la gestione e la memorizzazione dei dati è il *file system* (la parte del sistema operativo che si occupa della definizione e gestione dei file immagazzinati sulla memoria di massa). In un sistema EDP ogni applicazione opera in modo del tutto indipendente (o quasi) dalle altre applicazioni, facendo uso dei propri dati e dei propri programmi (fig. 15).



LO SAI CHE

Anche se un sistema informativo può essere gestito senza l'ausilio di mezzi elettronici, da adesso in poi, quando parleremo di *sistemi informativi* (SI), intenderemo quelli gestiti tramite computer.

fig. 15 Applicazioni nell'ambiente EDP tradizionale

Questo modo di procedere comporta una serie di problemi.

1. Le varie applicazioni risultano isolate dalle altre facenti parte dello stesso sistema informativo. In particolare, questo risulta un problema quando è necessaria la condivisione di dati da parte di due o più applicazioni (per esempio i dati dei clienti per l'applicazione che gestisce gli ordini in un'azienda e quella di supporto al marketing) o quando un'applicazione utilizza i dati forniti da un'altra. Infatti, se due applicazioni differenti lavorano sullo stesso file, quando una di esse ha

bisogno di apportare modifiche alla struttura dei dati, per esempio deve aggiungere un nuovo campo in un record, sarà necessario modificare anche la descrizione dei dati interna all'altra applicazione, modificandone la struttura. Questo non è sempre agevole e porta a volte alla duplicazione dei file.

2. Gli stessi dati vengono ripetuti tante volte quante sono le applicazioni che devono usarli, con conseguente spreco di memoria e con il rischio di introdurre inconsistenze ed errori.
3. L'accesso ai dati può avvenire solo tramite le applicazioni, cioè programmi fatti realizzare da utenti specializzati con conseguenti limitazioni nelle possibilità di richiesta di informazioni. Se c'è necessità di ottenere risultati diversi da quelli previsti dall'applicazione, questo è possibile solo con notevole dispendio di tempo e denaro per realizzare i programmi specifici.
4. Esiste uno stretto legame tra livello logico (che cosa si vuole fare in un'applicazione) e fisico (come sono memorizzati e gestiti i dati) con conseguenti difficoltà, lunghezza e costo nella modifica delle applicazioni anche a fronte di variazioni lievi.

Nasce l'idea di avere un *unico contenitore di dati* a cui tutti i programmi e gli utenti possano accedere contemporaneamente (*accessi multipli*) e con linguaggi semplici e non specialistici (*diversità di linguaggi*). Deve essere garantita la sicurezza di questi dati quando utenti diversi accedono ai dati (*privacy*), per salvaguardare gli stessi dai problemi fisici (*integrità*).

Gli utenti devono essere in grado di conoscere quali sono i dati a loro disposizione ed essere garantiti sulla loro correttezza e consistenza. La struttura dei dati deve poi poter essere facilmente e velocemente modificata, senza dover variare tutti i programmi facenti parte dell'applicazione.

Questi obiettivi però non possono essere raggiunti facilmente tramite gli strumenti convenzionali: nasce quindi l'esigenza di nuovi strumenti, quali le basi di dati.

■ Basi di dati e DBMS

LO SAI CHE

Raccolte di dati censuari, registri di stato civile, libri parrocchiali, archivi mercantili sono esempi di basi di dati.

Una **base di dati** può essere definita come una collezione di dati strutturati, progettati per essere usati in applicazioni differenti e da differenti utenti. In particolare, è un insieme di dati memorizzati senza ridondanze inutili per servire più di un'applicazione in contemporanea e organizzati in modo da essere indipendenti dai programmi che li usano.

Per la gestione delle basi di dati e la verifica della loro coerenza e consistenza sono necessari particolari sistemi software, chiamati DBMS (Data Base Management System) che si occupano della memorizzazione, dell'organizzazione e della gestione dei dati; quindi, è a essi che fanno capo tutte le operazioni di inserimento di nuovi dati, di cancellazione di quelli inutili, di modifica di quelli obsoleti e di ricerca. Un DBMS, quindi,

si colloca tra i programmi applicativi e i file dove sono memorizzati i dati e si occupa di gestirli su richiesta dei programmi stessi (fig. 16).

Con il vecchio sistema, ogni programma doveva interfacciarsi con il *file system*, mentre ora è solo il DBMS che lo utilizza. Nella fig. 17 viene evidenziato il modello di funzionamento di un sistema di tale tipo. Gli utenti accedono ai programmi tramite interfacce.

I programmi quando hanno bisogno di dati li richiedono al DBMS, che li preleva dalla base di dati tramite il *file system* del sistema operativo, restituendoli ai programmi richiedenti.

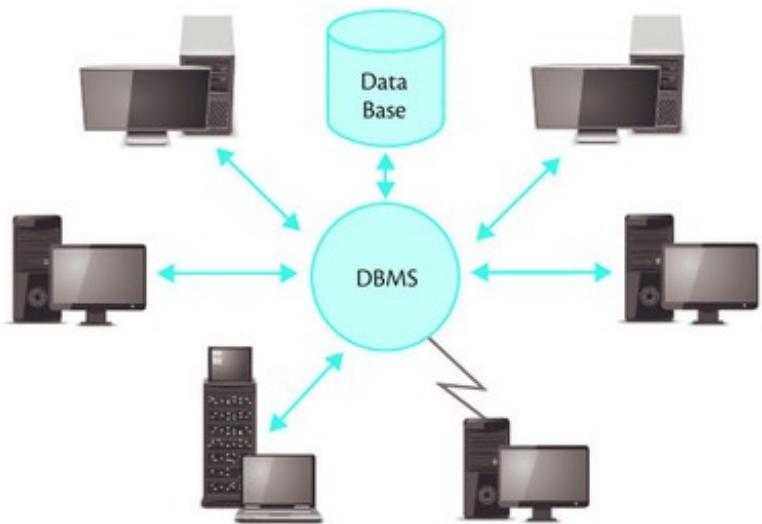


fig. 16 Base di dati

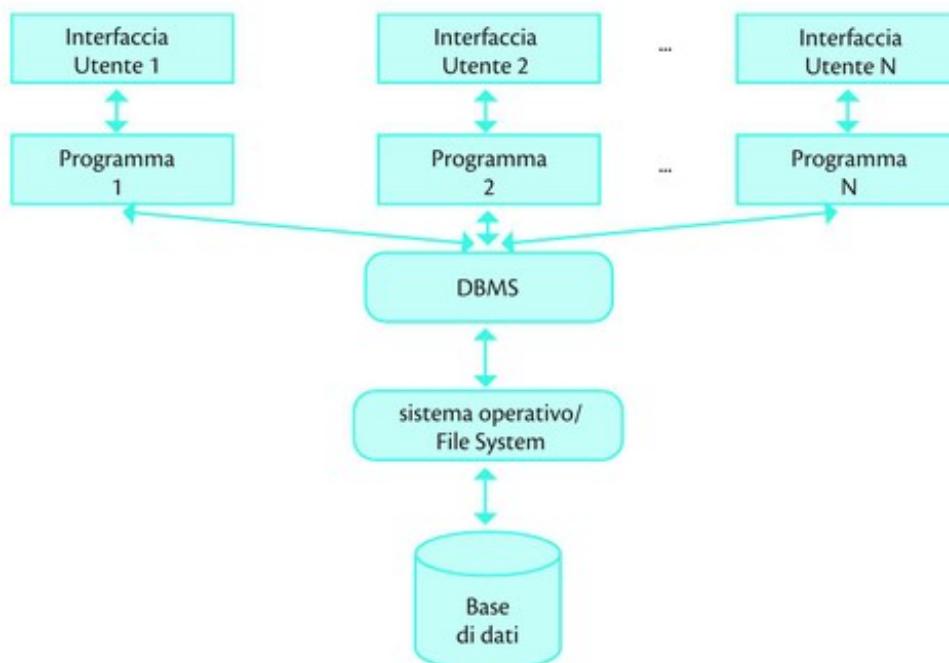


fig. 17 Sistema basato su DBMS

■ Vantaggi nell'uso dei database

Vediamo quindi quali sono i principali vantaggi nell'uso delle basi di dati rispetto all'uso dei file.

Eliminazione delle ridondanze

Proprio perché alla base di dati possono accedere applicazioni differenti, è possibile evitare la duplicazione delle informazioni. Dati uguali di differenti applicazioni non vengono duplicati se non nei casi in cui tutto il sistema ne trae un reale beneficio.

Per **ridondanza** si intende sia la duplicazione del dato, sia la memorizzazione di un dato che deriva dall'elaborazione di altri.

Per esempio, il numero di studenti di una classe costituisce una ridondanza se sono già stati memorizzati i nominativi degli studenti: basterebbe contare gli studenti stessi per sapere quanti ce ne sono nella classe. Tuttavia, se questa richiesta viene fatta spesso, può essere conveniente memorizzare questo dato (ridondanza utile).

Ciò comporta un costo maggiore, sia per l'occupazione di memoria sia per la sua gestione, ma vengono velocizzate le richieste.

È il responsabile della base di dati (DBA, DataBase Administrator) che deve decidere se inserire o no le ridondanze utili per garantire la massima efficienza della base di dati.

Eliminazione delle inconsistenze

Si parla di **inconsistenza** quando due dati che rappresentano la stessa informazione assumono valori diversi.

Per esempio, se l'indirizzo di un cliente è memorizzato in due file differenti può capitare che un'eventuale modifica sia apportata a un solo file; in questo caso si ha un'inconsistenza, perché si hanno due indirizzi diversi per lo stesso cliente.

Dal momento che la base di dati riduce le ridondanze, il cambiamento del valore di un dato viene immediatamente reso disponibile a tutti gli utenti, cioè *non possono esistere dati uguali con differenti valori*. Se per esempio viene modificato l'indirizzo di un cliente, il nuovo indirizzo sarà reso immediatamente disponibile a tutti.

Integrità dei dati

Il DBMS si occupa di controllare che l'inserimento di nuovi dati o la cancellazione di quelli già esistenti non alteri la congruenza della base di dati. I dati vengono protetti da accidentali o voluti cambiamenti che risultino non corretti o non autorizzati. Consideriamo il caso in cui la base di dati di una biblioteca sia costituita da due file, uno contenente i dati anagrafici degli autori e l'altro contenente quelli relativi alle loro opere: la cancellazione di un autore deve essere impedita, se ci sono ancora sue opere memorizzate nell'altro file. Anche l'inserimento di un'opera deve essere impedito se non è stato inserito l'autore.

FISSA LE CONOSCENZE

1. Che cos'è un sistema EDP?
2. Che cosa si intende per applicazione informatica?
3. Quali esigenze hanno portato alla nascita delle basi di dati?
4. Utilizzando un DBMS, come avviene l'accesso ai dati?
5. Quando una ridondanza può essere utile?
6. In che modo un DBMS garantisce l'integrità dei dati?

5. ARCHITETTURA

■ Il modello ANSI/SPARC

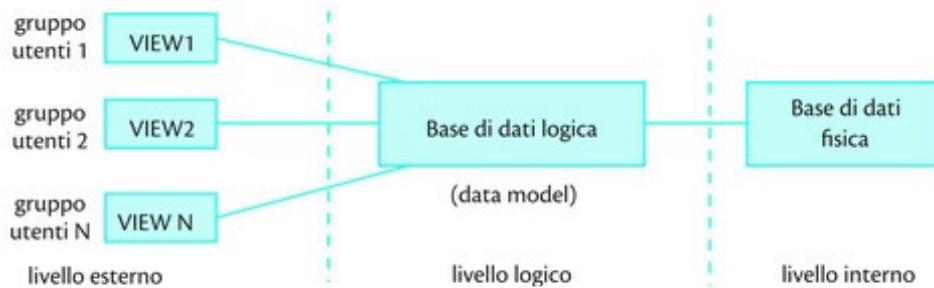
In un DBMS i dati si possono vedere, o meglio descrivere, a livelli diversi. Questa possibilità permette di liberare, da una parte il progettista della base di dati, dall'altra il programmatore delle applicazioni, dalla preoccupazione di come i dati sono fisicamente organizzati in memoria di massa, concentrandosi sul loro significato piuttosto che sulla loro organizzazione fisica.

Nell'organizzazione delle basi di dati si evidenziano tre livelli di astrazione (*esterno, logico, interno*), che permettono al DBMS di nascondere la reale organizzazione dei dati nei supporti di memorizzazione. Tutto ciò è ottenuto grazie a una particolare architettura, che è stata formalizzata da un organismo di standardizzazione (ANSI/SPARC) nel 1975. A questi tre livelli corrispondono i seguenti schemi di rappresentazione dei dati.

Sottoschemi o viste: *livello esterno* in cui differenti utenti possono avere differenti visioni degli stessi dati a seconda dell'uso che devono farne.

Schema logico: rappresenta la visione complessiva del database e consiste nella traduzione dello schema concettuale nel modello di rappresentazione dei dati adottato dal DBMS a disposizione. I moderni DBMS mettono a disposizione dei programmatori opportuni linguaggi, che permettono di creare schemi logici di database con semplici istruzioni ad alto livello.
Schema fisico: si pone al livello interno e descrive come sono organizzati fisicamente i dati sui supporti di memoria di massa.

La [fig. 18](#) descrive i livelli di astrazione secondo l'architettura ANSI/SPARC delle moderne basi di dati.



LO SAI CHE

ANSI

È l'organismo di standardizzazione americano (American National Standards Institute) e molti dei suoi standard sono stati accettati anche a livello internazionale.

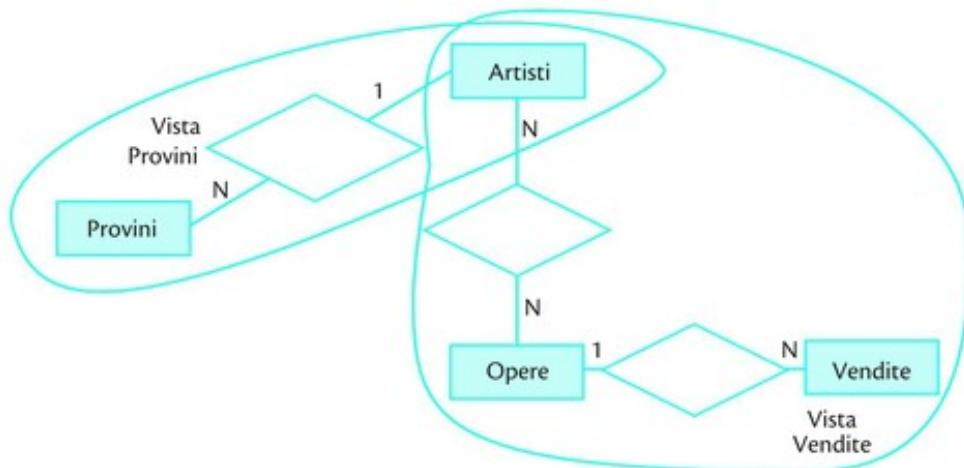
Livello esterno

È detto anche *livello applicativo* e descrive i dati come sono visti da una o più applicazioni, o da uno o più gruppi di utenti. Possono esistere più schemi esterni (o sottoschemi) ognuno riguardante la "vista" di interesse per una specifica applicazione. Si tratta di una parzializzazione dello schema concettuale, che permette di isolare le applicazioni da modifiche apportate ai livelli interno e concettuale. Semplifica il lavoro agli utenti, consentendo loro di ignorare ciò che non interessa e offrendo loro, nel contempo, una rappresentazione più familiare e conveniente dei dati.

fig. 18 Livelli di astrazione

Nell'esempio della [fig. 19](#) viene mostrata la gestione di una casa discografica. Si può "vedere" il reparto relativo alle vendite, oppure "vedere" quello relativo ai provini e ai relativi artisti ([fig. 19](#)). Ogni applicazione avrà libero accesso ai dati della propria "vista".

[fig. 19](#) Viste differenziate di uno stesso schema



Livello logico

Describe formalmente tutti gli oggetti di interesse per le applicazioni, offrendo una rappresentazione precisa e dettagliata delle strutture dati (record, campi, chiavi) necessarie per memorizzare le informazioni del sistema informativo. Permette sia l'aggiunta o la modifica di sottoschemi senza impatto sullo schema concettuale sia le modifiche allo schema concettuale in modo trasparente al livello esterno (programmi applicativi). Offre uno strumento di controllo sul contenuto e sull'utilizzo del database.

Livello interno

Rappresenta la descrizione formale delle strutture fisiche degli archivi che costituiscono la base di dati. È orientato all'efficienza e risente di valutazioni di tipo tecnico/economico (rapporto costo/prestazioni) relative alle tecnologie hardware e software utilizzate. Essendo legato alla tecnologia, deve poter cambiare in accordo con le evoluzioni tecnologiche, per conservare l'economia del sistema senza coinvolgere lo schema concettuale e tanto meno gli schemi esterni.

■ Indipendenza logica e fisica

Grazie all'organizzazione vista nel paragrafo precedente, è possibile capire meglio i concetti di indipendenza logica e fisica già accennati.

Per **indipendenza logica** si intende la possibilità di modificare lo schema logico, senza dover modificare i programmi che usano le singole applicazioni.

Tale indipendenza viene realizzata tramite il concetto di sottoschema o vista (view) descritto in precedenza. In alcuni casi, quando viene inserito un nuovo campo o ne viene eliminato uno, la modifica può interessare

solo alcuni sottoschemi e perciò solo in questo caso i programmi che li utilizzano andranno modificati. Tutti gli altri programmi non risentono della suddetta modifica, perché l'informazione cambiata non appartiene al loro sottoschema. Questo concetto implica l'esistenza di un insieme di programmi (facenti parte del DBMS) che effettuano la conversione di una richiesta espressa in termini di sottoschema in una espressa in termini di schema.

Per **indipendenza fisica** si intende la possibilità di modificare l'organizzazione fisica dei dati, senza dover modificare l'organizzazione logico-concettuale.

L'indipendenza fisica si basa sul concetto che né lo schema né il sottoschema rispecchiano il modo in cui i dati sono effettivamente memorizzati nella memoria secondaria del sistema.

Lo schema rimane invariato quando vengono utilizzate nuove tecniche di accesso ai file, quando una parte dei dati viene spostata da un supporto di memoria a un altro, quando vengono scelte diverse tecniche per il trattamento degli indici, quando vengono utilizzate nuove funzioni di hash e così via. Per esempio, si può avere la necessità di registrare i dati su supporti diversi, anche se questi devono essere considerati come un'unica unità.

Nella base di dati fisica, i dati sono memorizzati in file separati, ma quando ci sarà la necessità di prelevarli, il DBMS li cercherà indifferentemente sull'uno o sull'altro supporto. Se, in un secondo tempo, sarà necessario unificare i dati su un unico supporto, lo schema logico e le applicazioni non risentiranno del cambiamento. Anche in questo caso è compito del DBMS convertire le richieste, fatte in termini di schema, in chiamate alle routine del sistema operativo che permettono l'accesso ai dati. La [fig. 20](#) mostra dove interviene il DBMS e dove, invece, il sistema operativo nell'architettura delle basi di dati.

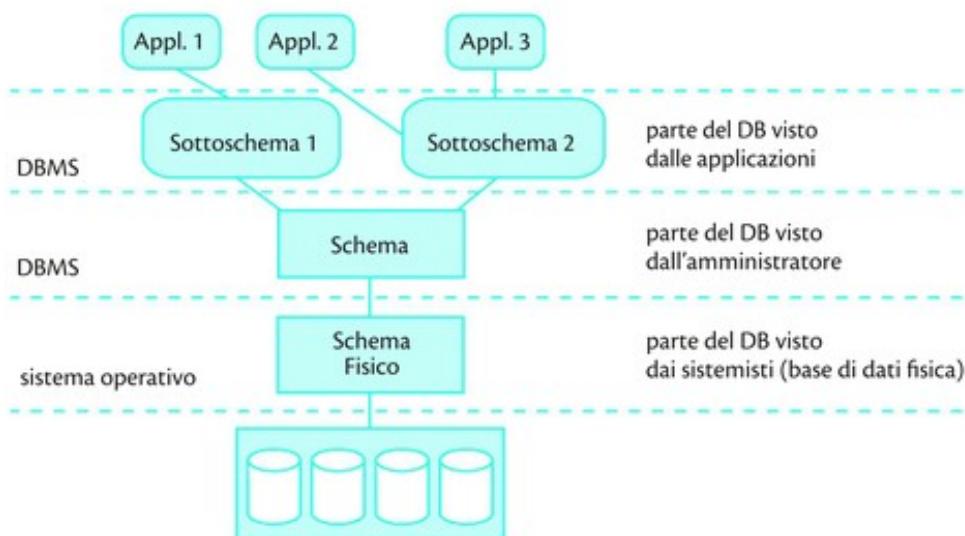


fig. 20 Architettura

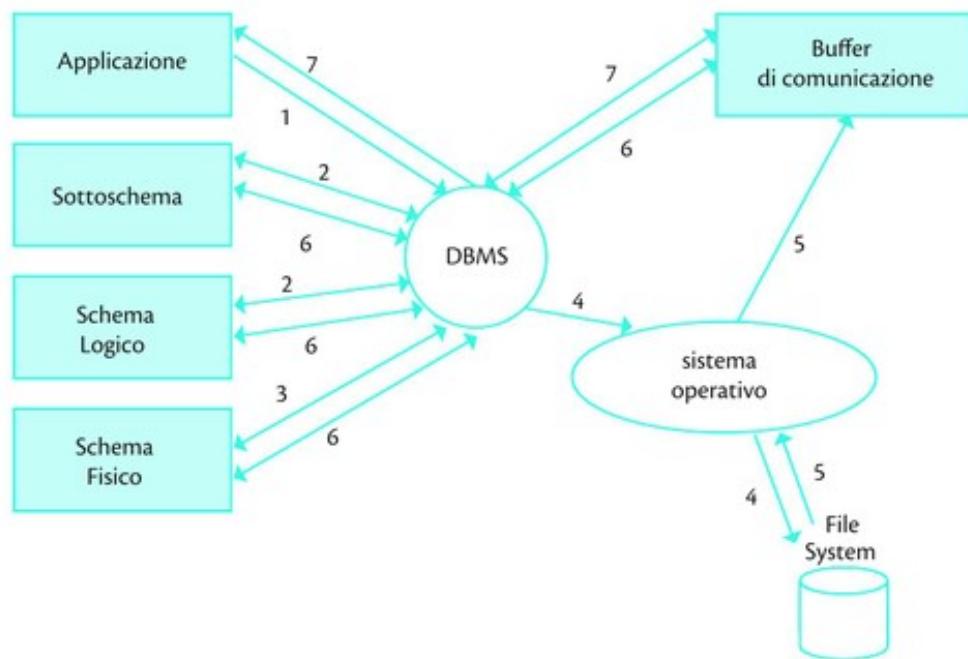
Il DBMS, quindi, gestisce tutti i passaggi tra i vari modi di vedere i dati, appoggiandosi sul SO. Provvederà anche al controllo sulla sicurezza e

sulla **privatezza** dei dati.

Per meglio capire come sono gestiti i dati e come funziona il DBMS, vediamo che cosa succede nel momento in cui un'applicazione necessita di un dato memorizzato nella base di dati. Per far questo ci facciamo aiutare dalla **fig. 21**.

1. Un'applicazione (o un utente) richiede un dato facente parte di una entità al DBMS (in modo esplicito o no), definendone alcune caratteristiche.
2. Utilizzando il sottoschema corrispondente all'applicazione in questione (modello esterno) il DBMS individua la corrispondenza tra l'entità richiesta e quella definita nello schema logico (corrispondenza esterno/logica).
3. Il DBMS, tramite lo schema fisico, individua dove il dato è fisicamente memorizzato (file) e i metodi per potervi accedere (corrispondenza logica/interna).
4. Il DBMS richiede al SO di prelevare il dato in questione dal file (o dai file).
5. Il SO restituisce, in un'apposita area, il record prelevato dal DBMS.
6. Il DBMS effettuerà le opportune trasformazioni (secondo le informazioni presenti nello schema fisico e in quello logico) del record reperito, per renderlo compatibile con il modello esterno.
7. Il DBMS restituirà un dato all'applicazione richiedente.

fig. 21 Passi svolti dal DBMS per reperire i dati



FISSA LE CONOSCENZE

1. Quali sono i tre livelli gestiti dal DBMS?
2. Che cos'è un sottoschema (o vista)?
3. Che cosa si intende per indipendenza logica?
4. Che cosa si intende per indipendenza fisica?
5. Come avviene lo scambio dei dati tra le applicazioni e il database?

6. LINGUAGGI E UTENTI

Linguaggi per i database

Per poter gestire le basi di dati tramite il DBMS si fa uso di due tipologie di linguaggi: i DDL e i DML. È da considerare che in alcuni ambienti di sviluppo vi è un solo linguaggio che copre entrambi gli aspetti, ma è sempre possibile distinguere le funzionalità dell'uno o dell'altro. Normalmente esiste la possibilità di fornire i comandi in modalità testuale, oppure tramite interfaccia grafica.

DDL

Per la definizione dello schema logico si usano i linguaggi **DDL** (Data Definition Language), che consentono di definire i tipi di entità presenti nello schema concettuale e le loro relazioni. Un DDL è in genere un linguaggio a sé stante con sintassi e semantica proprie, ed è indipendente dalle applicazioni. Viene usato soprattutto dall'amministratore del sistema (DBA, Data Base Administrator).

Il DDL serve anche a definire le viste (sottoschemi), consentendo a ogni programmatore di un'applicazione di selezionare solo la parte dello schema che gli interessa, e a definire alcuni parametri qualitativi (tipi di organizzazione) e quantitativi (dimensioni) delle strutture fisiche di memorizzazione della base di dati.

```
Create table Prodotti  
  (Codice      char (7),  
   Descrizione char (25),  
   Categoria   char (20),  
   Quantità    Integer);
```

I principali DBMS (Oracle, DB2, MySQL, Access e così via) forniscono anche ambienti grafici per l'amministrazione del sistema e, in particolare, per la definizione delle tabelle.

Nella [fig. 22](#) è mostrato un esempio di come si può creare una tabella usando l'interfaccia di Access.

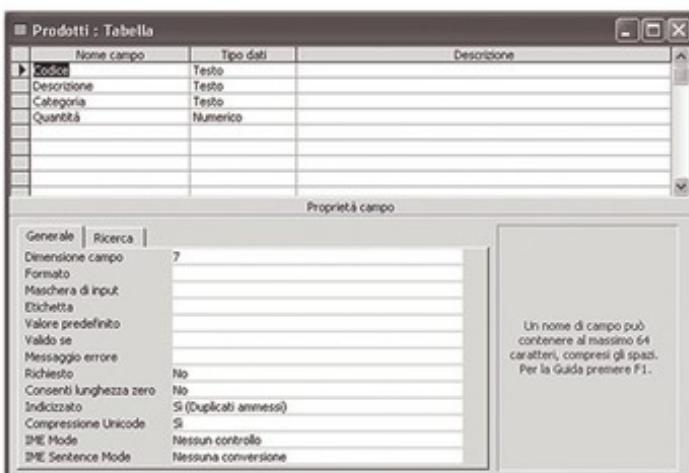


fig. 22 DDL con interfaccia grafica

DML

Per la modifica, il reperimento, l'inserimento e la cancellazione dei dati si usano i linguaggi di manipolazione dei dati (**DML**: Data Manipulation Language). Essi hanno lo scopo di fornire all'utente uno strumento per interrogare e modificare le informazioni contenute nella base di dati.

Un DML può essere un linguaggio a sé stante o un'estensione del linguaggio ospite, cioè un insieme di sottoprogrammi di libreria richiamati all'interno di un linguaggio di programmazione tradizionale (come C++, Visual Basic).

Nel primo caso siamo di fronte a un vero e proprio linguaggio di programmazione rivolto all'uso delle basi di dati, anche se spesso i sottoprogrammi vengono usati a loro volta all'interno di programmi scritti in un linguaggio ospite (per esempio istruzioni SQL possono essere richiamate all'interno di un programma Visual Basic).

Per quanto riguarda i DML, essi vengono distinti in:

- procedurali (o navigazionali) quando hanno gli operatori per trattare i singoli record;
- non procedurali (o dichiarativi) quando gli operatori sono indipendenti dal concetto di posizione, cioè considerano i dati collettivamente. Non si deve indicare la procedura da seguire (algoritmo), ma l'obiettivo da raggiungere.

Un particolare tipo di linguaggio di manipolazione è il *query language* (linguaggio di interrogazione). Si tratta di un linguaggio di tipo interattivo per l'interrogazione e la modifica della base di dati. È stato ideato principalmente per risolvere il problema degli utenti occasionali che, pur non avendo conoscenze di programmazione, possono interrogare la base di dati senza dover scrivere programmi in un linguaggio complesso. I sistemi attuali vanno sempre più nella direzione di dotarsi di strumenti grafici per agevolare l'attività di interfacciamento con l'utente.

Di seguito è mostrato un esempio in cui si richiede di reperire alcune informazioni della tabella Prodotti, utilizzando uno di questi linguaggi (SQL).

fig. 23 Query tramite interfaccia grafica



```
Select Codice, Descrizione  
From Prodotti  
Where Quantità < 100;
```

Vediamo come la stessa richiesta (query) possa venire formulata tramite un *query language* grafico (fig. 23).

Gli utenti

In base alle modalità di uso della base di dati possiamo individuare alcune categorie di utenti.

Il DBA (Data Base Administrator) è il responsabile del sistema per quanto riguarda sia la gestione dello schema e dei sottoschemi (creazioni e modifiche), sia per quanto riguarda l'organizzazione fisica dei dati. Ha infatti il compito di:

- creare inizialmente lo schema logico e successivamente mantenerlo;
- definire e mantenere lo schema interno;
- definire e aggiornare i diritti di accesso;
- ripristinare la base di dati in caso di malfunzionamento.

In **fig. 24** viene mostrato un esempio del pannello di controllo che utilizza il DBA per creare un database.

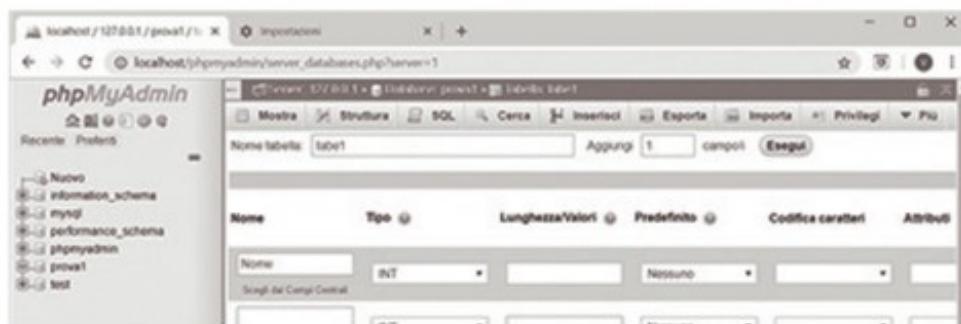


fig. 24 Pannello di controllo

Vi sono poi i *programmatori*, cioè coloro che realizzano le applicazioni utilizzando un DML o particolari linguaggi di programmazione.

Per ultimo abbiamo gli utenti finali. Questi, non avendo normalmente conoscenze informatiche, possono accedere direttamente ai database tramite interfacce amichevoli che li guidano nelle interrogazioni.

La maggior parte degli utenti è però inconsapevole di stare usando un database. Infatti tutti coloro che utilizzano delle applicazioni, anche in rete, non si rendono conto che nella maggior parte dei casi accedono a un database per recuperare i dati. Quando, per esempio, consultiamo un orario ferroviario, compriamo un biglietto per il cinema o la cassiera del supermercato passa la merce sotto lo scanner ed emette lo scontrino, noi utilizziamo (direttamente o indirettamente) una applicazione che accede a un database.

FISSA LE CONOSCENZE

1. Quali operazioni sono consentite dal DDL?
2. Quali operazioni sono consentite dal DML?
3. Quali sono i compiti del DBA?
4. Quali sono gli utenti delle basi di dati e con quali linguaggi vi accedono?

7. SICUREZZA NELLE BASI DI DATI

■ Problematiche di sicurezza

Le problematiche di sicurezza sui dati hanno un'importanza strategica all'interno delle aziende, anche se si sono sempre scontrate con l'esigenza della velocità di accesso.

Il controllo degli accessi, infatti, può incidere in maniera significativa sulla necessità di avere tempi di risposta efficienti (performance). Più si controlla in modo accurato e meno tempo il sistema potrà dedicare a rispondere alle richieste degli utenti. Una accurata gestione della sicurezza porta quindi a un notevole incremento della complessità di un sistema di elaborazione dati, riducendone l'efficienza in termini di tempi di risposta. Un DBMS è disegnato per risolvere entrambe queste esigenze con efficienza.

Le problematiche della sicurezza possono essere viste sotto l'aspetto della privatezza e sotto l'aspetto dell'integrità e della consistenza.

■ Privatezza

Il problema della privatezza dei dati contenuti in un database è estremamente importante. Nella banca dati di un'azienda è presumibile che non tutti gli utenti possano accedere a tutti i dati, ma solo a quelli di loro competenza. Il problema è ancora più grave in presenza di archivi anagrafici di vario tipo, perché questi consentono un elevato controllo sulle persone mediante, per esempio, la creazione di dossier personali. In ogni caso è indispensabile garantire il corretto e veloce reperimento delle informazioni solo alle persone autorizzate, impedendo l'accesso ad altri non autorizzati.

Questo problema presenta due principali aspetti.

1. Occorre stabilire un regolamento di privatezza in cui il Data Base Administrator determina quali utenti siano autorizzati ad accedere a quali dati e con quali operazioni (solo lettura, solo scrittura e così via). Il Data Base Administrator utilizza un sottoinsieme del linguaggio di definizione dei dati (DDL) per descrivere le leggi che devono proteggere la base di dati e che determinano chi è autorizzato a svolgere certe funzioni e in quali circostanze.
Oltre alle informazioni, bisogna proteggere anche lo schema della base di dati, cioè la descrizione dei dati stessi con tutte le regole di privatezza a essi associate.
Il regolamento di sicurezza viene fornito agli utenti insieme allo schema.
2. Occorre controllare che questo regolamento venga rispettato. A ogni richiesta il DBMS verifica gli attributi del richiedente con le condizioni del regolamento.

Nell'esempio dalla **fig. 25**, la tabella degli accessi prevede per l'utente Rossi la possibilità di modificare (diritto RW) i PRODOTTI e di leggere solamente (diritto R) gli ORDINI. Quindi se effettua una operazione di lettura su ORDINI (caso a) gli verrà fornito l'insieme dei dati richiesti. Se invece cerca di modificare un ORDINE (caso b), riceverà un rifiuto. Nel caso in cui il DBMS decida che la richiesta non debba essere soddisfatta, la violazione, o meglio il tentativo di violare una delle regole, può venire notificato al Data Base Administrator e contemporaneamente venire imposta una "penale" al richiedente: abort del programma, scollegamento del terminale, eliminazione dell'utente dall'elenco degli autorizzati al sistema e così via.

I livelli ai quali i meccanismi di controllo possono agire sono molteplici e dipendono dal grado di sofisticazione del DBMS.

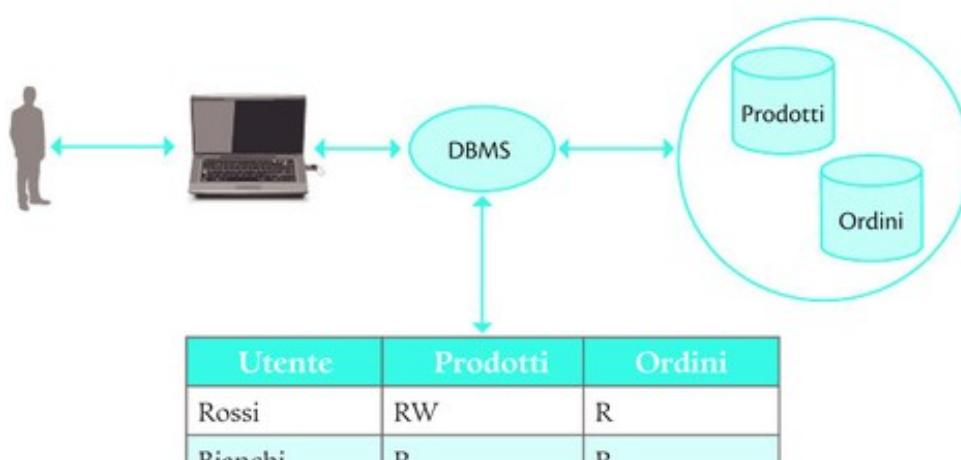
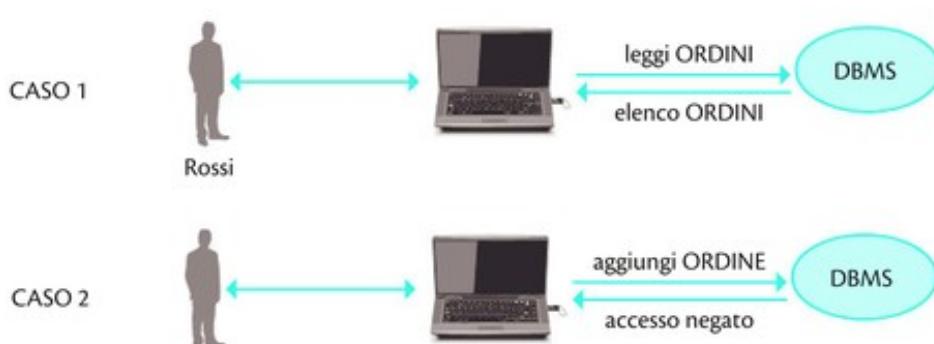


fig. 25 La privatezza dei dati



Vincoli di integrità

L'integrità logica consiste nella necessità di preservare la struttura logica della base di dati, ovvero le relazioni esistenti tra i dati. Questo è un requisito fondamentale per il successo del sistema informativo.

La gestione di tutte le informazioni, che sono una parte integrante del patrimonio dell'azienda, deve essere sicura e corretta, altrimenti le conseguenze possono essere disastrose per un'organizzazione altamente informatizzata.

Un aspetto fondamentale è la protezione contro l'aggiornamento errato della base di dati.

LO SAI CHE

Definiamo come **vincolo** una proprietà che deve essere soddisfatta da tutte le istanze corrette della base di dati.

Per poter svolgere questo compito, al DBMS vengono fornite alcune regole da rispettare per validare l'inserimento o la modifica di un dato nel database. Queste regole possono essere semplici **vincoli intra-relazionali**, se sono definiti sugli attributi di una sola entità (vincoli di unicità, vincoli di dominio, limitando per esempio il campo di esistenza di un campo) oppure più sofisticati vincoli inter-relazionali, se sono definiti su più entità contemporaneamente tenendo conto di interrelazioni tra i campi anche di archivi diversi (integrità referenziale).

I **vincoli di integrità** vengono così definiti dall'amministratore della banca dati, codificandoli in uno schema che viene fornito a tutti i software applicativi che richiedono l'accesso alla base di dati.

Vincoli di dominio

I vincoli di dominio sono i più semplici, e definiscono in genere il campo di esistenza, o range di dominio, di un attributo. Esprime quindi condizioni sul valore assunto da un singolo attributo. Se per esempio stiamo definendo i dati relativi a persone, un possibile vincolo sarà quello di non permettere di inserire delle età negative. Quindi il vincolo potrà essere:

Età > 0

Può essere anche un'espressione booleana (and, or, not) di predicati semplici. Per esempio il voto in un compito in classe dovrà essere compreso tra due estremi:

Voto > 0 and Voto <= 10

Vincoli di relazione

In questo caso si esprimono condizioni sui valori assunti tra campi legati tra loro. Può essere un'espressione booleana (and, or, not) di predicati semplici (confronto tra attributi, tra attributi e costanti, ...). Per esempio, per un Prodotto il costo netto e il costo con IVA devono essere legati, per cui il vincolo potrà essere:

Prezzo = Costo + PerclVA * Costo

Il vincolo può anche essere condizionale, per cui:

Tipopatente deve essere NotNull se Patente ='SI'
LODE non può esserci se il VOTO è inferiore a 30

Nella tabella seguente di studenti universitari la prima e ultima riga non possono essere accettate, perché nel primo caso il voto è fuori range (vincolo di dominio), mentre nel secondo caso la lode può essere assegnata solo se il voto è 30 (vincolo di relazione).

Matricola	Studente	Voto	Lode
54646	Rossi	33	NO
56432	Bianchi	28	NO
33668	Verdi	30	SI
43546	Gialli	27	SI

Vincoli di integrità referenziale

Nel caso di vincoli inter-relazionali, vanno controllate le situazioni in cui i dati risultano correlati, anche se presenti su archivi diversi (*integrità referenziale*). Se abbiamo per esempio un database di una banca con dati relativi ai conti correnti e alle operazioni su di essi, bisogna evitare che venga fatto un movimento su un conto corrente inesistente. Il DBMS blocca quindi l'inserimento di un movimento (per esempio un prelievo di denaro) se esso è relativo a un conto corrente bancario non ancora registrato a meno che, con opzioni specifiche, non venga indicato di inserire anche il conto corrente nell'archivio (vedi il successivo concetto di transazione). In modo analogo il DBMS impedirà di cancellare un conto quando ci sono ancora dei movimenti di valuta a esso collegati. Anche in questo caso è possibile specificare al DBMS di proseguire la cancellazione del conto, eliminando anche tutti i movimenti a esso collegati.

LO SAI CHE

Non è possibile demandare questi compiti di controllo interamente ai programmi applicativi, sia per non appesantire eccessivamente il software, sia perché diverse applicazioni possono avere la necessità di gestire lo stesso dato, generando così delle incongruenze.

Consistenza della base di dati

La coerenza dei dati è collegata al controllo di validità ed è legata al fatto che taluni aggiornamenti richiedono la modifica contemporanea di dati correlati.

La contemporaneità dell'operazione di modifica di più di un dato è solo un'impressione che viene data all'utente della banca dati. Le operazioni vengono in realtà eseguite in maniera strettamente sequenziale ed è necessario che durante il lasso di tempo che intercorre tra l'inizio della prima modifica e la fine dell'ultima tutti i dati interessati nell'operazione rimangano "congelati" per tutti gli altri utenti.

Il concetto di **transazione** è particolarmente importante, quindi cerchiamo di comprenderlo bene. Una transazione è un insieme di istruzioni che formano un'unità di lavoro indivisibile. La transazione inizia con la base di dati che si trova in uno stato consistente e quando termina deve lasciare la base di dati in uno stato consistente. All'interno di una transazione la base di dati può essere non consistente.

Una base di dati si dice **consistente** quando tutti i vincoli di validità dei dati sono rispettati.

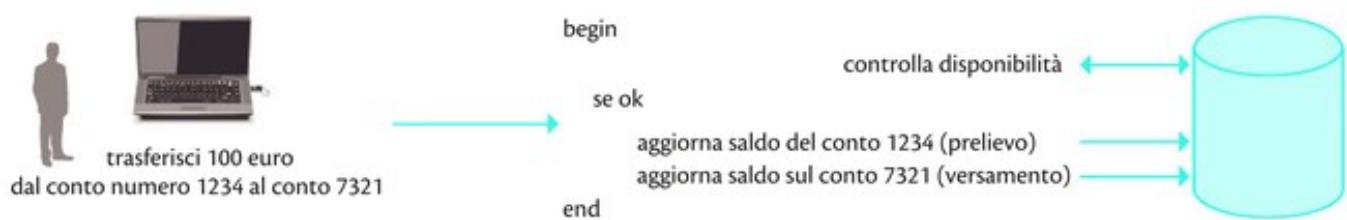
Le modifiche effettuate all'interno di una transazione non sono visibili alle transazioni concorrenti, finché tutte le modifiche della transazione non sono state completate. Per fare questo all'interno di una transazione il DBMS porrà dei *lock* (*blocchi*) sui dati da modificare e li rilascerà a fine transazione. Un *lock* è un meccanismo che permette a una sola transazione per volta di modificare i dati.

Ogni transazione è encapsulata tra i due comandi `Begin_transaction`, che contrassegna il punto di inizio, ed `End_transaction`, che indica la fine della transazione. Tra questi due comandi sono poste le istruzioni di lettura e/o scrittura della base di dati. Solo le transazioni che terminano normalmente effettuano il *commit* confermando il lavoro svolto sulla base di dati e facendola transitare in un nuovo stato.

LO SAI CHE

Esempi di transazione sono l'aggiornamento di un conto corrente, il pagamento di un bonifico bancario, una prenotazione aerea, l'acquisto di un biglietto. Gli utenti interagiscono con la base di dati attraverso programmi applicativi i quali usano le transazioni per garantire la correttezza del dato inserito/letto.

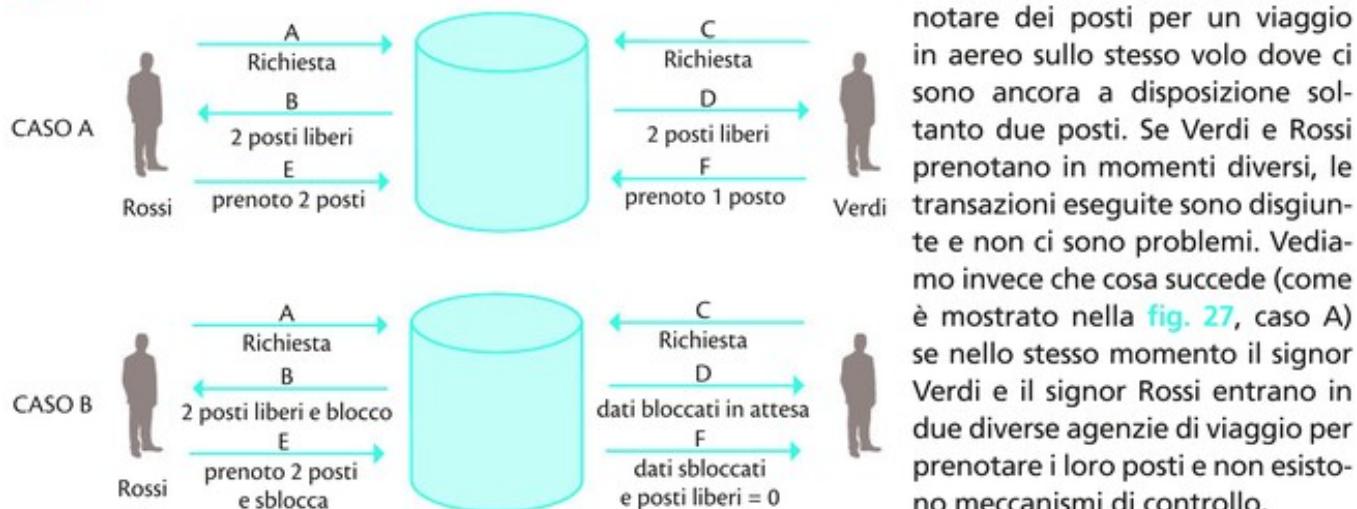
fig. 26 Esempio di transazione



Le transazioni che, invece, terminano prematuramente se si verifica un qualche errore (hardware o software) durante l'esecuzione sono trattate come se non fossero mai iniziate, perché effettuano il rollback riportando la base di dati alla situazione precedente la transazione.

Si consideri per esempio la transazione per il pagamento di un bonifico (fig. 26) bancario, che consiste nel trasferimento di una somma di denaro da un conto corrente a un altro: è evidente che non è ammessa l'esecuzione parziale della transazione, per esempio solo il prelievo da un conto e non il versamento sull'altro conto; inoltre l'importo versato deve essere identico a quello prelevato, ossia al termine della transazione la somma dei due saldi non deve cambiare. In assenza di circostanze strane la transazione, dopo l'aggiornamento della base di dati, effettua il commit, rendendo valide e permanenti le modifiche apportate.

fig. 27 Accessi concorrenti



- Rossi chiede quanti posti liberi ci sono.
- Il sistema fornisce il valore di Postiliberi = 2 e si mette in attesa della risposta di Rossi per sapere se vuole prenotare o no.
- Verdi richiede quanti posti liberi ci sono.
- Anche questa richiesta fornisce il valore di Postiliberi = 2, poiché al momento esistono ancora 2 posti liberi. Anche in questo caso il sistema si mette in attesa della risposta di Verdi per sapere se vuole prenotare o no.
- Rossi decide di prenotare e occupa 2 posti, e il sistema decremente il numero di posti, ponendo Postiliberi = 0.
- Anche Verdi prenota, ma è ancora convinto che Postiliberi sia uguale a 2, quindi prenota un posto, e il sistema decremente il numero di posti, ponendo Postiliberi = 1.

Al momento dell'imbarco ci sarà una spiacevole sorpresa, che sarebbe stata evitata se l'aggiornamento del record con l'informazione sui posti fosse stato effettuato tenendo conto della possibile concorrenza di accesso da parte di più transazioni. Per non creare conflitti è necessario applicare il principio della mutua esclusione: un insieme di dati può essere utilizzato da una sola transazione per volta, e solo quando l'uso da parte di quella transazione è terminato, può essere riutilizzato da altre. Quando una transazione effettua una richiesta di accesso, questo viene bloccato tramite un meccanismo di lock, e rilasciato (unlock) solo al termine di tutte le operazioni.

Vediamo (caso B della fig. 26) cosa succederebbe in questo caso nell'esempio precedente.

- Rossi chiede quanti posti liberi ci sono.
- L'informazione di Postiliberi verrà bloccata quando viene inviata a Rossi l'informazione.
- Verdi chiede quanti posti liberi ci sono.
- Viene risposto a Verdi che i dati sono bloccati e lo si mette in attesa.
- Dopo che Rossi avrà terminato la prenotazione, il sistema porrà Postiliberi = 0, e i dati verranno rilasciati.
- Verrà comunicato a Verdi (che era in attesa) che Postiliberi = 0. Quindi Verdi non potrà prenotare un posto ormai inesistente.

FISSA LE CONOSCENZE

- Che cosa garantisce la privatezza?
- Che cosa garantisce l'integrità?
- Quando una base di dati si dice consistente?
- Che cosa si intende per transazione?
- Come viene garantita l'integrità fisica?
- Quando due transazioni sono concorrenti?
- Come agiscono i meccanismi di lock e unlock?
- Quale funzione ha il rollback?



RIPASSIAMO INSIEME

I dati in azienda

Nelle aziende **dati e informazioni** sono gestiti tramite i **sistemi informativi automatizzati** (ICT), che vanno dalle semplici applicazioni su database a sistemi complessi in grado di gestire e integrare i dati di tutta l'azienda.

Memorizzare dati

La conservazione dei dati avviene tramite i file, che possono essere **testuali, binari o strutturati**.

Nei file strutturati le informazioni sono organizzate in **record** sui quali è possibile effettuare operazioni sia logiche sia fisiche. I record possono essere **logici** (come vengono visti e definiti dal programmatore) e **fisici** (il blocco di dati gestito dall'elaboratore). I file **sequenziali** sono archivi in cui i dati sono registrati in sequenza. Non è possibile accedere direttamente a un record di un file sequenziale.

File ad accesso diretto

Un file si definisce **ad accesso diretto** quando è possibile accedere a un record tramite il valore di una chiave. Questo fatto è possibile nei file indexed tramite la memorizzazione delle chiavi nel file degli indici. Un file si definisce ad **accesso calcolato** quando tramite la chiave è possibile calcolare la posizione che il record occupa all'interno del file.

Dal File System alle basi di dati

L'uso dei file per gestire i sistemi informativi pone delle **limitazioni** e rende la gestione dei dati **più complessa**. Una base di dati è una raccolta di dati strutturati, progettata per essere accessibile da parte di applicazioni diverse e differenti utenti. Il **DBMS** (Data Base Management System) è l'insieme di strumenti software che permettono a uno o più utenti

di acquisire e/o modificare il contenuto della base di dati. Un DBMS offre i seguenti vantaggi: meccanismi di **protezione** di accesso ai dati, **riduzione** delle ridondanze e delle inconsistenze, **sicurezza** e **ottimizzazione** nell'uso dei dati.

Architettura

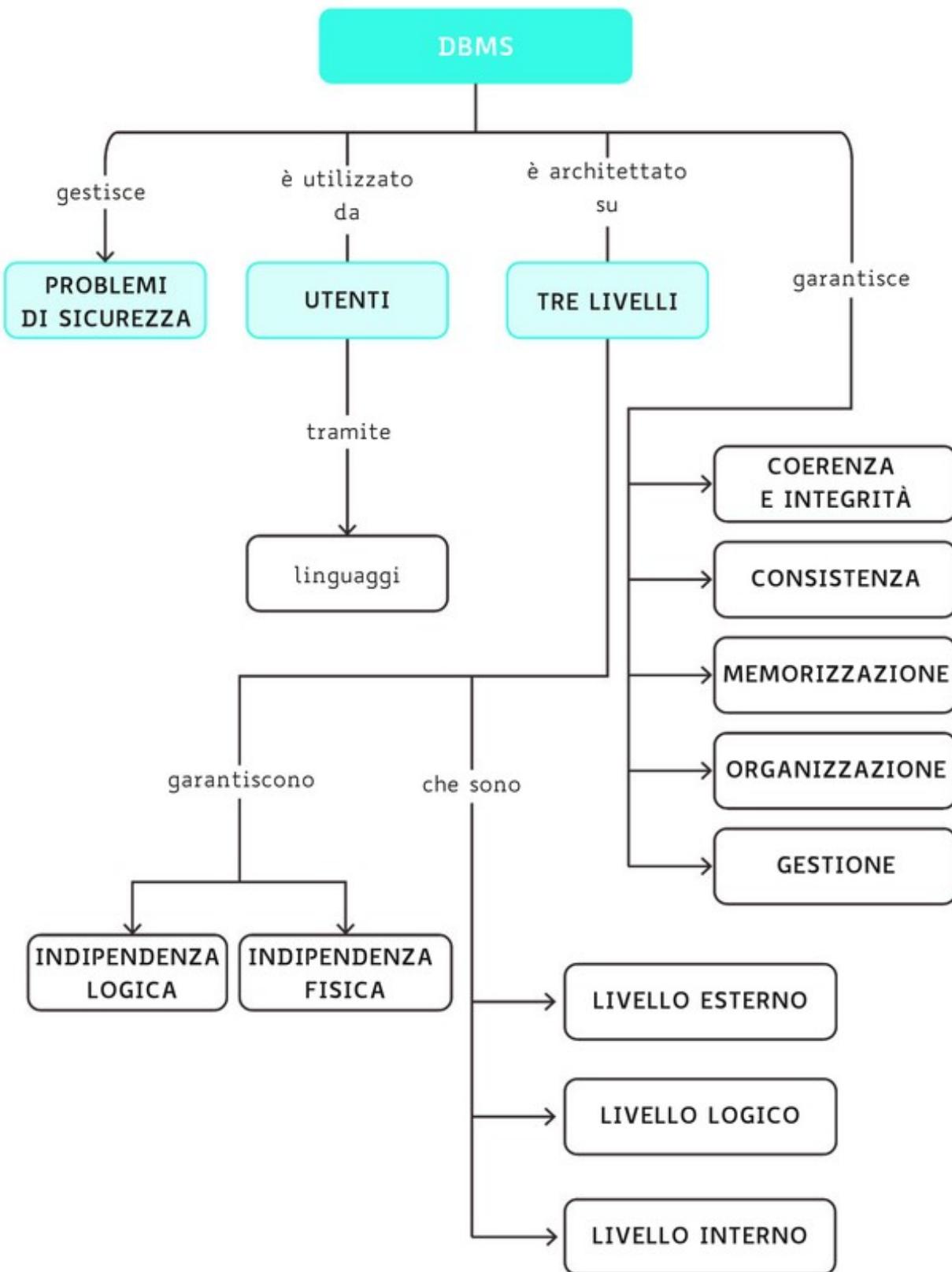
L'architettura di un DBMS prevede tre livelli: il livello **esterno**, che è quello vicino al singolo utente, in quanto corrisponde al modo di vedere e organizzare i dati dell'utente stesso, quello **logico**, che rappresenta l'organizzazione logica dei dati nel DBMS, quello **interno**, che corrisponde al modo secondo il quale i dati sono realmente memorizzati. Questa architettura realizza l'**indipendenza logica e fisica**.

Linguaggi e utenti

Un linguaggio per la **definizione dei dati** (DDL, Data Definition Language) è un linguaggio che permette di descrivere gli oggetti contenuti nella base di dati. Un linguaggio per la **manipolazione dei dati** (DML, Data Manipulation Language) consente all'utente di interrogare il database, di inserire, di modificare e di cancellare le informazioni. Gli utenti di un database possono essere suddivisi in **tre classi**: il DBA (Data Base Administrator), i programmatore e gli utenti finali.

Sicurezza nelle basi di dati

Le problematiche di sicurezza di una base di dati hanno un'importanza strategica in azienda. Il DBMS deve garantire l'**integrità dei dati**, consentendo l'aggiornamento solo ad utenti autorizzati, la **protezione dal danneggiamento** (sia da azioni illegali, sia da cause di forza maggiore), e il **controllo degli accessi** concorrenti, consentendo la multiutenza.



TEST**TEST**

Svolgi il test interattivo

Vero o falso?

1. Record logico e record fisico sono sinonimi.
2. L'operazione di lettura comporta il trasferimento di un campo del record della memoria di massa alla memoria centrale.
3. Con l'organizzazione sequenziale è possibile accedere direttamente a un record del file.
4. È possibile ordinare un file sequenziale a indice su più chiavi.
5. Database e DBMS sono sinonimi.
6. Una ridondanza può causare un'inconsistenza.

Scelta multipla (una sola è la risposta esatta)

7. ICT vuol dire:
 - A Information Conventional Transported.
 - B Informazione Comune Tipizzata.
 - C Information Communication Technology.
 - D Internet Center Technology.
8. DBMS significa:
 - A Data Base Management Server.
 - B Data Base Manipulation Server.
 - C Data Base Management System.
 - D Data Base Manipulation System.
9. Con il termine ridondanza si indica:
 - A un programma.
 - B un dato memorizzato nel database.
10. DML significa:
 - A Data Modern Language.
 - B Data Manipulation Library.
 - C Definition Model Language.
 - D Data Manipulation Language.
11. Una base di dati è:
 - A un insieme di programmi.
 - B un insieme di dati strutturati.
 - C un insieme di routine per la gestione dei file.
 - D un insieme di interfacce grafiche.

PREPARATI PER IL COLLOQUIO ORALE

1. Che cosa si intende per sistema informativo? [vedi lez. 1]
2. Quali sono i differenti tipi di file? [vedi lez. 2]
3. A che cosa serve il file degli indici? [vedi lez. 3]
4. Qual è la definizione completa di base di dati? [vedi lez. 4]
5. Che cosa si intende per accessi concorrenti? [vedi lez. 7]
6. Quali vantaggi offre l'indipendenza logica? [vedi lez. 5]
7. Chi sono gli utenti di un database? [vedi lez. 6]
8. Quali sono i livelli del modello ANSI/SPARC? [vedi lez. 5]



CLIL – IN ENGLISH, PLEASE



AUDIO

Ascolta la pronuncia
del testo

ABSTRACT

Database

A file is a collection of informations related to objects of the same type, stored on a permanent storage medium. Each element of the file is called record. The organization of a file can be sequential, indexed, hash, relative.

The old information systems where based on the use of files, but the inter-dependency of programs and data created many problems.

A database is a collection of data stored, avoiding redundancies, which is normally used for more than one application. DBMS is the set of software which enables users to interact with the database. The DBMS has to guarantee the integrity of the data by allowing only authorized users to update it and by controlling access to their database. The language defining the structure of the data is the DDL (*Data Definition Language*) and the language allowing data to be manipulated is the DML (*Data Manipulation Language*). The users of a database are the DBA (*DataBase Administrator*) who defines the structure of the data, the programmers who write the programs with the instructions which enable access to the database and the end users who use the database.

EXERCISES

True or false?

1. In a sequential file you cannot delete records in the middle of the file. T F
2. A database is made up of files. T F
3. Database and DBMS are synonymous. T F
4. A redundancy may cause an inconsistency. T F
5. Data may be modified by using DDL. T F
6. DBMS allows access. T F

Multiple choice

7. DBMS stand for:
 - A Data Base Management Server
 - B Data Base Management System
 - C Data Base Manipulation Server
 - D Data Base Manipulation System
8. DDL stand for:
 - A Data Description Language
 - B Data Definition Language
 - C Description Data Language
 - D Definition Data Language
9. If you access directly to a record using a key, you are using:
 - A a relative file
 - B a sequential file
 - C a hash file
 - D a binary file

GLOSSARY

- Database:** total data stored, without redundancy, which is used for one or more applications.
- DBA (Data Base Administrator):** person responsible for maintaining a database.
- DDL (Data Definition Language):** language for defining data.
- DML (Data Manipulation Language):** language used to manipulate the content of the database.

GLOSSARIO
CLIL

2

Progettare una base di dati



ESERCIZI COMMENTATI

Segui la risoluzione passo passo



PREREQUISITI

- Conoscere la programmazione procedurale.
- Conoscere gli aspetti base dell'analisi di un problema.

PER COMINCIARE

1. I dati di interesse nella gestione di una scuola sono:

- A studenti, classi, insegnanti
- B dipendenti, studenti, materie
- C dipendenti, auto, insegnanti
- D studenti, redditi, voti principali

2. Le caratteristiche principali di un'auto sono:

- A proprietario, cilindrata, prezzo
- B targa, modello, marca
- C colore, velocità, anno
- D dimensione, colore, n. posti

3. L'indipendenza logica serve per:

- A modificare la struttura dei dati
- B modificare i programmi
- C modificare la struttura dei dati senza modificare tutti i programmi
- D modificare i file

CONOSCENZE

- Conoscere le principali fasi della progettazione di un database.
- Conoscere le caratteristiche del modello concettuale E/R.
- Conoscere i principali tipi di relazioni.
- Conoscere il concetto di gerarchia e i principali tipi.

ABILITÀ

- Saper effettuare la programmazione concettuale usando il modello E/R.
- Saper stabilire associazioni tra le entità.
- Saper ritagliare una vista su di uno schema.

COMPETENZE

- Utilizzare le strategie del pensiero razionale negli aspetti dialettici e algoritmici per affrontare situazioni problematiche elaborando opportune soluzioni.
- Scegliere dispositivi e strumenti in base alle loro caratteristiche funzionali.
- Gestire progetti secondo le procedure e gli standard previsti dai sistemi aziendali di gestione della qualità e della sicurezza.

1. LA PROGETTAZIONE DI UN DATABASE

Dati e informazioni

Prima di affrontare il tema della progettazione delle basi di dati, ricordiamo alcuni concetti importanti relativi ai dati e alle informazioni.

Per **dato** si intende un fatto raccolto tramite osservazioni e/o misurazioni, mentre per **informazione** si intende l'interpretazione e/o il collegamento tra i dati che può permettere di prendere decisioni.

Per poter gestire i dati e le informazioni, in un'azienda si fa uso del **sistema informativo**, cioè di un mezzo per raccogliere, organizzare, immagazzinare e correlare dati e per estrarre e distribuire informazioni. Il sistema informativo (SI) non richiede necessariamente l'uso di strumenti elettronici e/o informatici, e in esso viene rappresentata solo la realtà funzionale al raggiungimento di obiettivi predeterminati (**fig. 1**).



fig. 1 Rapporto tra sistema informativo e realtà

Un **modello** rappresenta una particolare percezione di una parte della realtà e il processo di modellizzazione serve per fissare e rappresentare questa percezione (**fig. 2**). Nel **processo di modellizzazione** noi selezioniamo determinati aspetti e li astraiamo per poter comprendere più semplicemente i fenomeni che vogliamo analizzare.

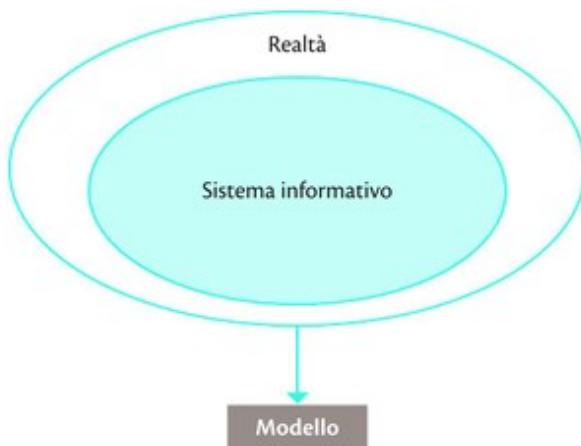


fig. 2 Processo di modellizzazione

Il SI non rappresenta quindi la realtà aziendale, ma il **modello** di una parte della stessa.

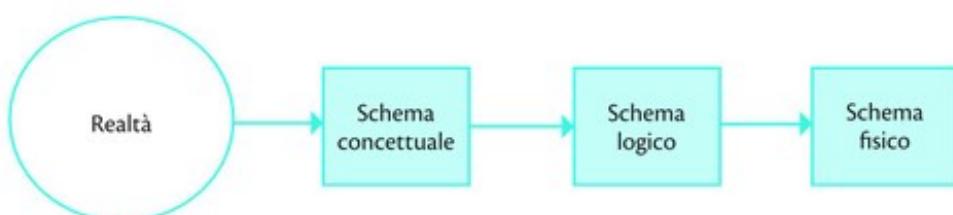
■ Fasi della progettazione

Il compito primario del progettista di una base di dati (DBA, *Data Base Administrator*) è quello di decidere quale parte della realtà è di interesse per le applicazioni che utilizzeranno la base di dati stessa.

Esistono alcune **metodologie** per progettare una base di dati: alcune specifiche per particolari tipi di problemi o valide per particolari modelli di dati; altre, più generali, valide per qualsiasi tipo di problema. Ricordiamo comunque che in generale una metodologia non è altro che la formalizzazione del buon senso con cui normalmente i progettisti lavorano.

Un modello abbastanza diffuso è quello descritto brevemente di seguito, che suddivide la progettazione in quattro fasi (fig. 3).

fig. 3 Fasi di progettazione



1. Raccolta e analisi dei requisiti

Attraverso colloqui e contatti di ogni genere (interviste, questionari) gli utenti forniscono al progettista i **requisiti del sistema informativo** in termini di dati, processi, vincoli. Le informazioni acquisite vengono descritte in appositi glossari di dati e informazioni.

2. Progettazione concettuale

A partire dai requisiti forniti nella fase precedente si modellano delle descrizioni formalizzate di come gli utenti vedono i dati a partire dalle operazioni ("viste di utenti"). Le viste vengono poi integrate in un unico **schema concettuale**.

Lo schema è l'insieme dei dati che fanno parte della base di dati e delle relazioni che intercorrono tra di essi.

3. Progettazione logica

Lo schema concettuale viene trasformato in uno **schema logico**. Questa fase, a differenza della precedente, **tiene conto del modello su cui si basa il sistema di gestione scelto** (DBMS) che può essere relazionale (il più diffuso) oppure reticolare e gerarchico (storicamente il primo ad apparire sul mercato, ma più complesso da realizzare).

4. Progettazione fisica

Vengono definite le strutture di memorizzazione (tipi di dischi) e i parametri di tali strutture (ampiezza dei blocchi), tenendo conto delle caratteristiche del DBMS scelto.

FISSA LE CONOSCENZE

1. Che differenza c'è tra dato e informazione?
2. Quali sono i compiti del DBA nella fase di progettazione?
3. Quali sono le fasi per progettare un database?

2. IL MODELLO E/R - ENTITÀ E ATTRIBUTI

In una base di dati la descrizione dei dati avviene a livello astratto, indipendentemente dal computer e dal software usato.

Esistono vari modi per descrivere i dati in modo astratto e indipendente dalla macchina:

- i **modelli logici** sono il risultato di un compromesso tra l'esigenza di modellare facilmente la realtà e quella di disporre di un'implementazione efficiente. È la modalità con cui i DBMS gestiscono i dati;
- i **modelli semantici**, chiamati anche **modelli concettuali**, sono dotati di meccanismi di astrazione e di operatori. Hanno il dichiarato scopo di permettere la modellazione della realtà secondo una logica naturale per l'uomo e vengono tipicamente utilizzati nella fase di progettazione.

Lo schema Entità/Relazioni

Per la descrizione concettuale dei dati si usa un modello semantico basato sull'individuazione e la definizione delle entità di interesse e delle associazioni esistenti tra di esse.

Si tratta del **modello Entità/Relazioni** (*E/R, Entity/Relationship*) che, pur essendo semplice e intuitivo, è completo dal punto di vista semantico e può essere facilmente descritto graficamente (fig. 4).

a) Rappresentazione dell'entità



b) Rappresentazione dell'associazione



c) Rappresentazione degli attributi



fig. 4 Rappresentazione delle informazioni nel modello E/R

In questo modo si può modellare e descrivere la realtà, individuando le entità che ne fanno parte e le possibili relazioni (o associazioni) tra di esse.

Entità

Le **entità** sono elementi della realtà che, essendo dotati di caratteristiche comuni, vengono raggruppati in una **classe**.

Si chiama **istanza** (o **occorrenza**) dell'entità un esemplare della classe; viene determinata nel momento in cui vengono qualificate le caratteristiche dell'entità.

Per esempio, se Persona è un'entità con caratteristiche nome e città, il signor Rossi di Milano è un'istanza della classe.

Un'entità viene rappresentata graficamente con un rettangolo.

Attenzione! In molti casi si usa però il termine **entità** per indicare sia la classe sia un'istanza, rischiando di generare quindi confusione; di solito se ne comprende comunque il significato dal contesto.

Attributi

Gli **attributi** sono le proprietà attraverso cui è possibile descrivere un'entità (o un'associazione).

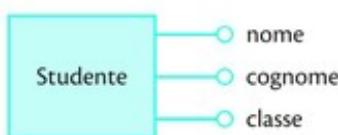
Vengono rappresentati graficamente con dei **punti collegati alle entità o alle associazioni**. Non possono esistere da soli e sono identificati dal nome, dal tipo (numero, stringa, carattere, data, ecc.) e dal dominio, cioè dall'insieme dei valori che possono assumere (per esempio "M" o "F" per l'attributo sesso).

A seconda del tipo di dato gli attributi possono essere di diversi tipi.

■ **Semplici** (o elementari): sono quelli che hanno un tipo semplice (per esempio interi, reali, stringhe, ecc.). Consideriamo per esempio l'entità Studente della [fig. 5](#), che descrive gli studenti iscritti in una scuola: i suoi attributi semplici sono nome, cognome, classe.

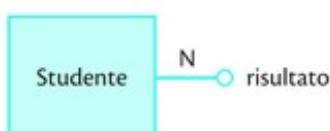
■ **Composti**: sono quelli in cui esistono sottoattributi componenti (per esempio una data, [fig. 6](#)); in pratica si tratta di insiemi di attributi che possono essere considerati come un'unica proprietà dell'entità. La data di nascita di uno studente è un attributo composto dell'entità Studente, perché può essere suddiviso in giorno, mese e anno di nascita. Un altro esempio di attributo composto è indirizzo, che contiene al suo interno via, numero e città.

Nello schema E/R gli attributi composti si rappresentano come nella [fig. 6](#).



[fig. 5](#) Attributi semplici

[fig. 6](#) Attributi composti



[fig. 7](#) Attributi multipli

■ **Multipli**: sono quelli di tipo "sequenza", in cui cioè esiste un numero variabile di sottoelementi componenti di uno stesso tipo (per esempio il risultato scolastico raggiunto nelle classi frequentate dallo studente). Nello schema E/R gli attributi multipli sono caratterizzati dalla presenza del simbolo N sull'arco che lega l'entità all'attributo ([fig. 7](#)).

Gli attributi possono essere opzionali o obbligatori. Nel primo caso si permette che all'attributo non vengano assegnati valori (per esempio, l'attributo fax di una persona può essere lasciato vuoto), mentre nel secondo l'attributo deve essere presente (per esempio, la data di nascita di una persona va obbligatoriamente specificata).

LABORATORIO

IL PROBLEMA Una casa discografica vuole memorizzare i dati relativi ai dischi prodotti e ai cantanti che incidono dischi con la casa discografica stessa. Individua le entità e le relazioni pertinenti.

L'ANALISI Nel problema proposto possiamo individuare due diverse **entità**: **Disco** e **Cantante**. Specifichiamo gli **attributi** di ciascuna entità, utilizzando le tabelle seguenti.

DISCO

CANTANTE

Nome attributo	Descrizione	Tipologia
codice	codice univoco per identificare il disco	semplice
titolo	Titolo dell'album	semplice
durata	Durata dell'album	composto (minuti, secondi)
data	Data di registrazione	composto (giorno, mese, anno)
brani	Brani inseriti nell'album	multiplo

Nome attributo	Descrizione	Tipologia
Nome d'arte	Nome d'arte con cui è noto	semplice
cognome	Cognome del cantante	semplice
nome	nome del cantante	semplice
indirizzo	Indirizzo di residenza	composto (via,numero civico, cap, città)
tel	Numeri di telefono su cui è reperibile	multiplo



FISSA LE CONOSCENZE

1. Che differenza c'è tra entità e istanza di entità?
2. Che cosa indica un attributo composto? E un attributo multiplo?

3. LE CHIAVI

Soltamente esiste un attributo (o insieme di attributi) che identifica univocamente le istanze di un'entità: questo attributo è la **chiave primaria**. Nello schema E/R si evidenzia sottolineando l'attributo.

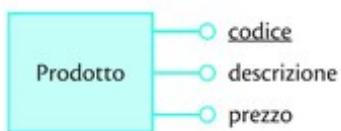


fig. 8 Chiave primaria

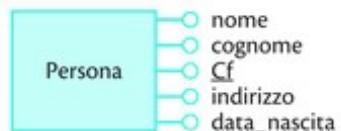


fig. 9 Entità Persona

Nel caso illustrato nella [fig. 8](#), per esempio, il codice "a21" individua univocamente un'istanza (un prodotto) dell'entità Prodotto. Non abbiamo bisogno di ulteriori indicazioni per sapere a quale prodotto stiamo facendo riferimento. In questo caso, nell'entità non possono esistere doppiioni, ovvero due o più istanze uguali, altrimenti la chiave primaria sarebbe duplicata e non permetterebbe di distinguere le istanze. Non saranno quindi ammessi prodotti con lo stesso codice.

Se nell'entità Persona scegliamo nome e cognome come chiave primaria, non saranno ammesse persone che hanno lo stesso nome e cognome. Se per esempio dovessimo realizzare la base dati dell'anagrafe italiana ([fig. 9](#)), non potremmo mai scegliere la coppia nome e cognome come chiave primaria, ma dovremmo invece ricorrere al codice fiscale che, essendo legato ai dati anagrafici, garantisce l'univocità per ogni persona.

Può capitare che in un'entità ci siano diversi attributi **candidati** a diventare chiave primaria. Nel diagramma E/R la scelta ricade sulla chiave che ha un **significato più specifico**. Per esempio, nell'entità Studenti di una università, gli attributi matricola e codice fiscale potrebbero essere entrambi usati come chiave primaria: tra i due, però, il numero di matricola è più legato al contesto universitario che stiamo modellando.

Generalmente i criteri per la scelta della chiave primaria sono i seguenti:

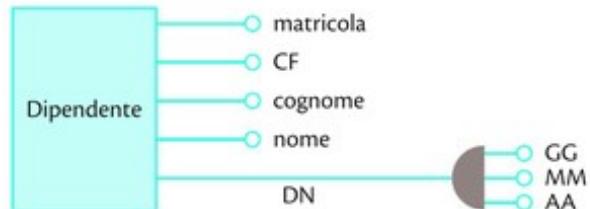
- si preferisce una chiave semplice a una composta;
- si tiene conto dei campi più utilizzati nelle richieste di accesso ai dati;
- si tiene conto del contesto del problema.

LABORATORIO

IL PROBLEMA Individuare le chiavi candidate dell'entità Dipendente e scegliere tra queste la chiave primaria, motivando la scelta.

L'ANALISI In questa entità le chiavi candidate potrebbero essere molteplici: matricola, codice fiscale (CF), l'insieme degli attributi cognome e nome (perché è abbastanza difficile che ci siano due dipendenti con lo stesso cognome e nome) oppure l'insieme degli attributi cognome, nome e data di nascita (DN). Per la scelta della chiave primaria dobbiamo tener conto del

problema che stiamo trattando. Visto che l'azienda assegna una matricola a ogni dipendente, è più ragionevole scegliere questo attributo come chiave primaria, perché presumibilmente la maggior parte delle richieste (*query*) saranno fatte partendo dal numero di matricola.



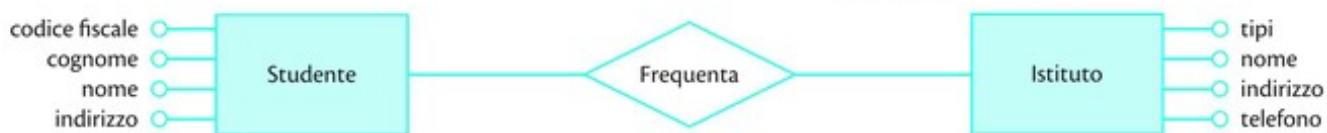
4. LE RELAZIONI 1:1 E 1:N

Le **relazioni** (o **associazioni**) sono i legami logici tra due o più entità. Corrispondono esattamente al concetto matematico di relazione tra elementi di un insieme. Anche in questo caso si può parlare di **classi di relazioni**, se si considerano tutte le associazioni possibili tra due entità.

La relazione viene rappresentata graficamente con un rombo collegato alle entità associate, e non può esistere slegata da esse.

Chiariamo meglio il concetto di relazione con un esempio e consideriamo le entità Studente e Istituto, le cui istanze rappresentano rispettivamente gli studenti della popolazione italiana e i vari istituti scolastici presenti sul territorio. Si intuisce facilmente che esiste una relazione tra le due entità che indica a quale istituto è iscritto uno studente. L'associazione viene rappresentata graficamente con un rombo collegato alle entità e viene identificata da un verbo che esprime un'azione (**fig. 10**).

fig. 10 Entità e relazioni



Tipi di relazioni tra entità

I tipi di relazioni tra entità riprendono i concetti tipici dell'insiemistica.

Relazioni 1:1 – A ogni elemento del primo insieme ne corrisponde **uno e uno solo** del secondo insieme e viceversa (per esempio l'associazione Coniugato tra Uomo e Donna della **fig. 11**).

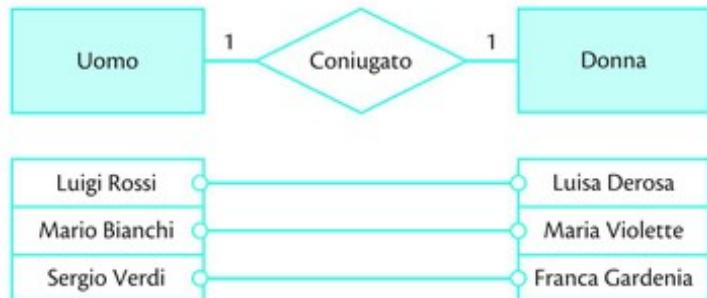


fig. 11 Relazione 1:1

Relazioni 1:N – A un elemento del primo insieme possono corrispondere **più elementi** del secondo insieme, ma non viceversa (per esempio l'associazione Frequenta tra Classe e Studente della **fig. 12**).

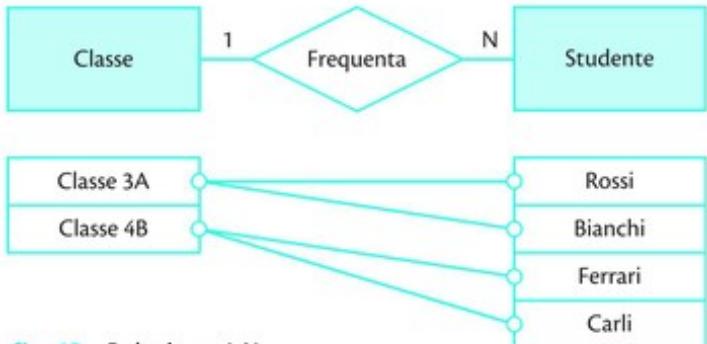


fig. 12 Relazione 1:N

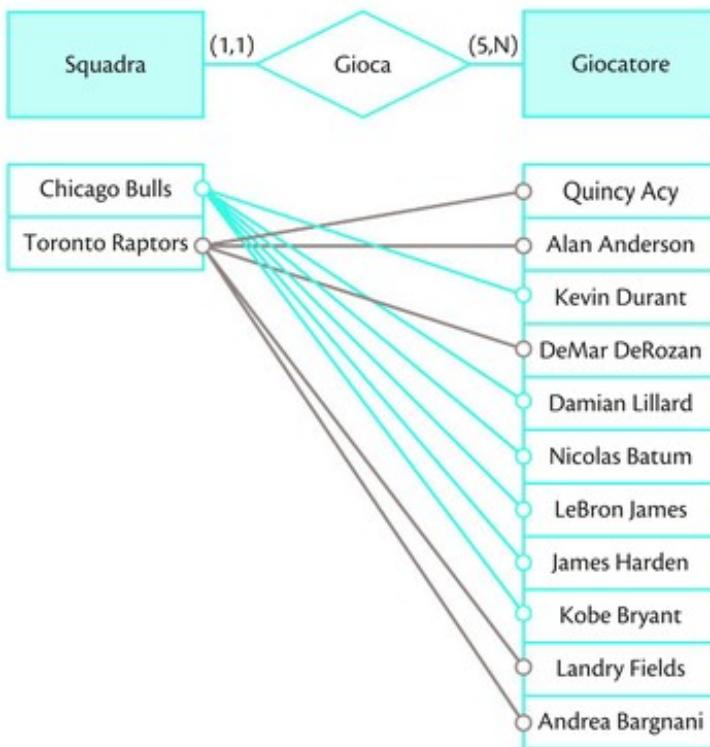


fig. 13 Cardinalità minima e massima

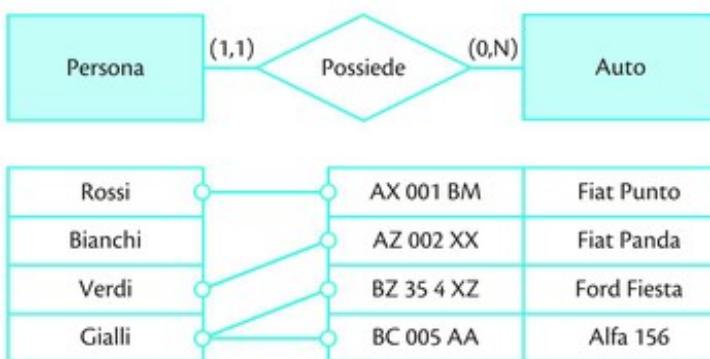


fig. 14 Associazione con partecipazione opzionale

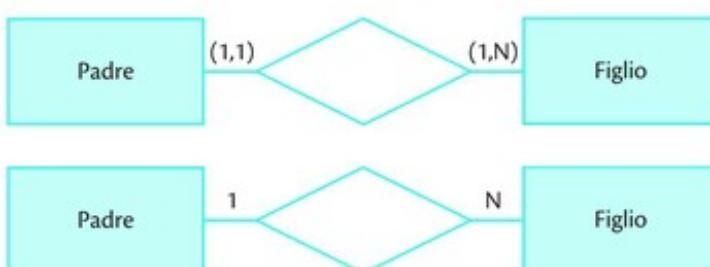


fig. 15 Due modi per rappresentare la cardinalità di un'associazione

Cardinalità delle associazioni

La cardinalità di un'associazione definisce il numero di istanze con cui l'entità può partecipare in un'associazione. Generalmente la cardinalità è identificata da due numeri, che definiscono la partecipazione minima e massima. Nella fig. 13 è rappresentata la situazione di una serie di squadre di basket: in ogni squadra devono essere presenti almeno 5 giocatori per poter giocare.

La cardinalità minima uguale a zero indica che l'entità partecipa in modo opzionale all'associazione. Nella fig. 14 è mostrata l'associazione opzionale Possiede tra l'entità Persona e l'entità Auto, in cui si prevede che una persona possa non possedere un'auto, ma che un'auto debba avere obbligatoriamente un proprietario.

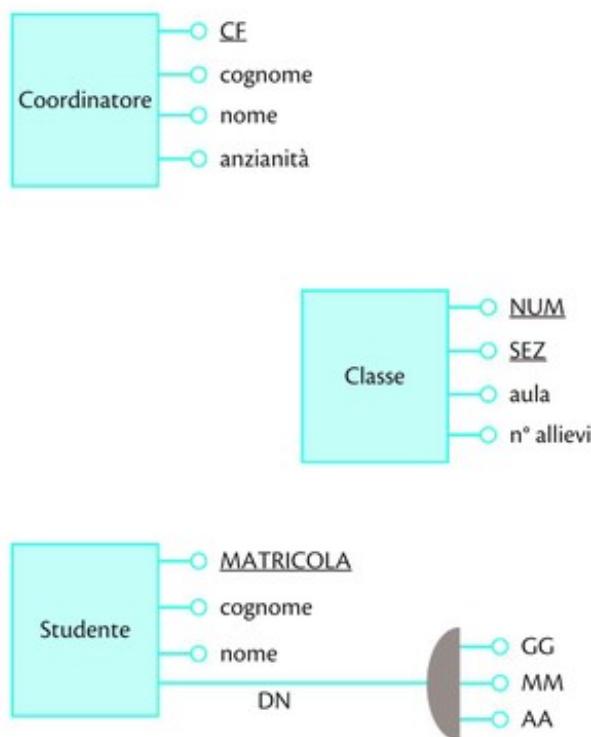
Se la cardinalità minima di un'associazione non viene rappresentata, si sottintende che è uguale a 1, come è mostrato nella fig. 15, dove Padre partecipa all'associazione con cardinalità (1,1) e Figlio con cardinalità (1,N).

LABORATORIO

IL PROBLEMA Realizzare lo schema E/R per la gestione di un istituto scolastico di cui si vogliono rappresentare i dati delle classi, degli studenti e degli insegnanti coordinatori.

L'ANALISI In questo problema possiamo individuare le entità: Coordinatore (con gli attributi CF, cognome, nome, anzianità), Classe (con numero, sezione, aula, numero allievi), Studente (con matricola, cognome, nome, data di nascita). Per l'entità Coordinatore sceglieremo come chiave primaria codice fiscale, per la Classe l'insieme degli attributi numero e sezione, per lo Studente matricola.

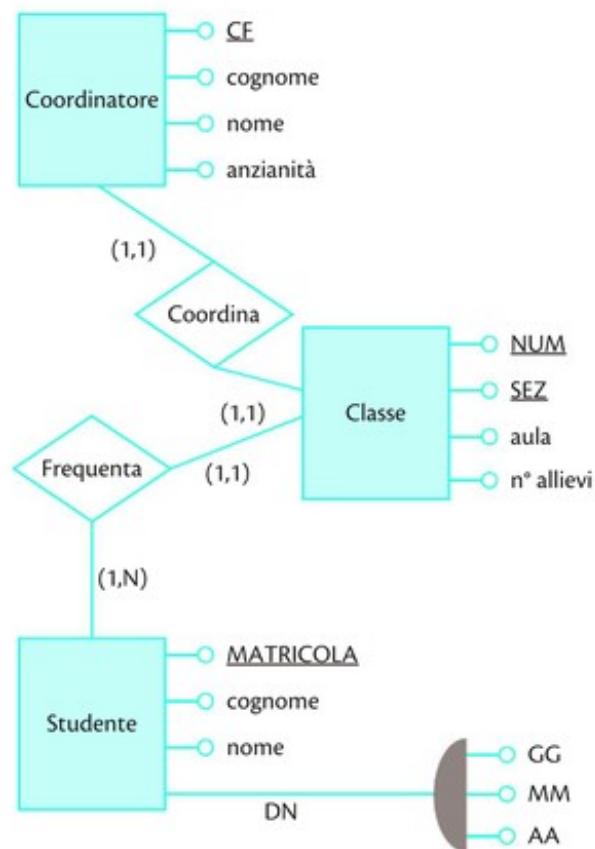
Lo schema E/R sarà il seguente:



Definiamo ora le relazioni tra le entità:

- tra Coordinatore e Classe vi è una relazione 1:1, poiché supponiamo che un coordinatore coordini una sola classe e una classe sia coordinata da un unico insegnante;
- tra Classe e Studente vi è una relazione 1:N, poiché una classe ha uno o più studenti, mentre uno studente appartiene a una e una sola classe.

Lo schema E/R completo sarà il seguente:



FISSA LE CONOSCENZE

1. Quando tra due entità esiste un'associazione di tipo 1:1?
2. Quando tra due entità esiste un'associazione di tipo 1:N?
3. Cosa indica la cardinalità di un'associazione?

5. LE ASSOCIAZIONI N:N E LE RELAZIONI CON ATTRIBUTI

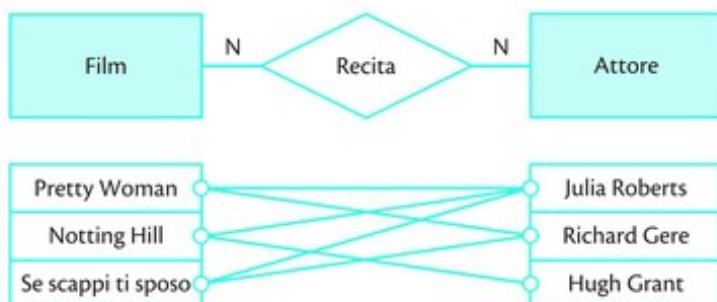


fig. 16 Relazione N:N

Le relazioni N:N sussistono tra entità che coinvolgono diverse istanze. Per usare il linguaggio insiemistico, nelle relazioni N:N a un elemento del primo insieme possono corrispondere più elementi del secondo insieme e viceversa (per esempio, l'associazione Recita tra Film e Attore della fig. 16).

Associazioni con attributi

Gli attributi possono essere riferiti sia alle entità sia alle associazioni quando specificano una caratteristica legata alla relazione esistente tra due entità. Per maggior chiarezza analizziamo subito un esempio (fig. 17).



fig. 17 Relazione con un attributo

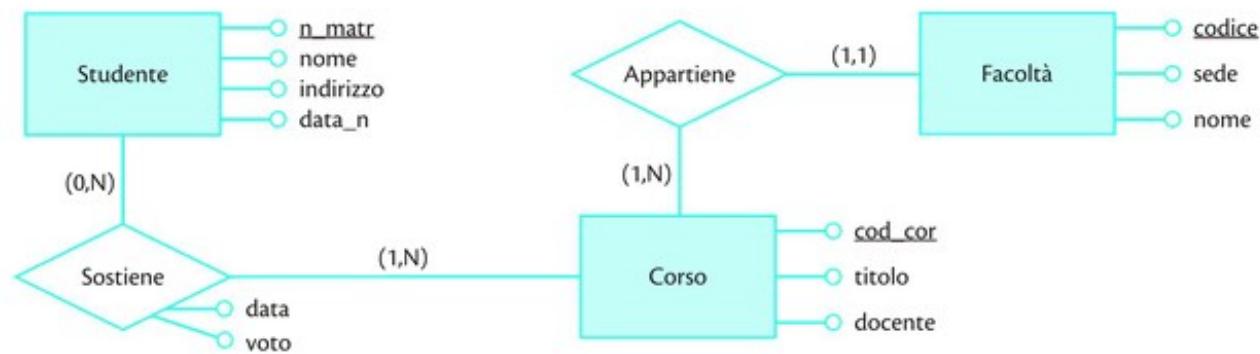
Consideriamo le entità Film e Attore: le istanze dell'entità Film rappresentano tutti i film esistenti nel campione che stiamo analizzando, mentre nell'entità Attore sono presenti le informazioni relative agli attori che hanno interpretato i film. Tra le due entità è presente una relazione di tipo N:N (Interpreta) perché un film è interpretato da molti attori e un attore può interpretare più film. Vogliamo ora registrare il ruolo che ha avuto un attore in un film (attore protagonista, secondo attore, comparsa, ecc.): risulta evidente che l'attributo ruolo non può essere legato all'entità Film (perché nel film sono presenti diversi ruoli) e neppure all'entità Attore (nella sua carriera l'attore avrà interpretato diversi ruoli). Ruolo è infatti un attributo della relazione Interpreta, perché dipende sia da Film sia da Attore e specifica il ruolo che ha avuto un attore in un determinato film.

LABORATORIO

IL PROBLEMA Nell'ambito della gestione degli esami universitari, considera i dati relativi alle facoltà (Informatica, Fisica, Lettere, ecc.), ai corsi, agli studenti e ai corsi superati dagli studenti con una prova d'esame. La base dati deve essere in grado di fornire l'elenco dei corsi presenti in una facoltà e la stampa delle informazioni (corso, voto e data) relative a tutti gli esami sostenuti da uno studente.

L'ANALISI La rappresentazione dei dati caratterizzanti la gestione degli esami universitari deve tenere conto del fatto che esistono più facoltà, ognuna delle quali attiva più corsi, che gli studenti devono superare sostenendo gli esami. Uno studente può aver superato zero o più esami e possono esserci più studenti che hanno superato l'esame di un corso.

Lo schema E/R risultante è il seguente:



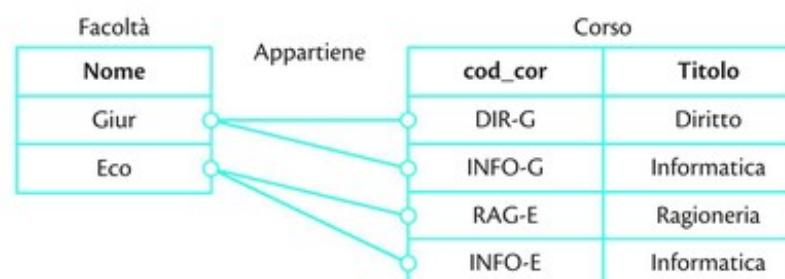
Nel seguito per la descrizione dello schema E/R usiamo delle tabelle dette dizionario entità e dizionario associazioni, leggiamo le associazioni presenti nello schema e per ogni entità definiamo gli attributi che la compongono con le loro caratteristiche (tipo, dimensione, vincoli).

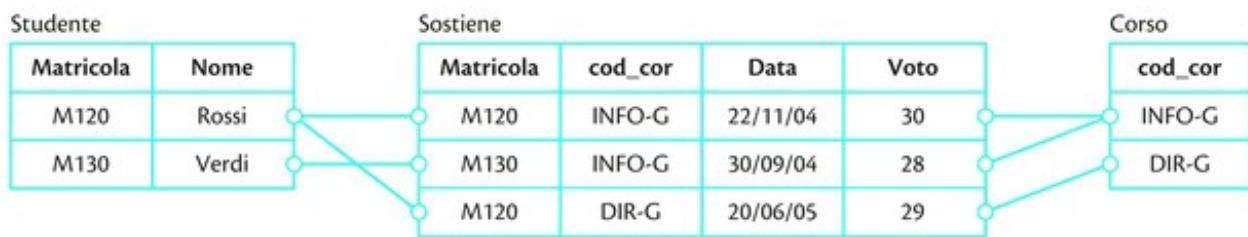
DIZIONARIO ENTITÀ

DIZIONARIO ASSOCIAZIONI

Nome	Descrizione	Attributi	Identificatore
universitaria possono appartenere corsi, ma un corso è relativa una sola facoltà.			
Tradicità Corso e l'entità Studente esiste la relazione molti a molti (Sostiene), poiché molti studenti sostengono l'esame per un corso e uno studente sostiene esami relativi a molti corsi. I dati relativi al voto e alla			
Corso	corsi attivati nelle varie facoltà	cod_cor titolo docente	cod_cor
STUDENTE FACOLTÀ CORSO	studenti che hanno sostenuto esami	n_matr nome indirizzo data_nascita	n_matr
L'associazione Sostiene ha i seguenti attributi:			

Nome	Descrizione	Entità coinvolte	Attributi
Appartiene	associa una facoltà a ciascun corso	facoltà (1,1) corso (1,N)	codice
Sostiene	associa un corso con gli studenti che hanno superato il relativo esame	corso (1,N) studente (0,N)	data voto





Nome attributo	Descrizione	Tipo	Lunghezza	Vincoli
n_matr	numero di matricola dello studente	stringa	8 byte	obbligatorio
Nome	nome e cognome	stringa	40 byte	obbligatorio
Indirizzo	indirizzo	stringa	40 byte	obbligatorio
data_n	data di nascita	data	8 byte	obbligatorio

Nome attributo	Descrizione	Tipo	Lunghezza	Vincoli
Codice	codice identificativo della facoltà	stringa	8 byte	obbligatorio
Sede	indirizzo e città dove ha sede la facoltà	stringa	50 byte	obbligatorio
Nome	denominazione della facoltà	stringa	20 byte	obbligatorio

Nome attributo	Descrizione	Tipo	Lunghezza	Vincoli
cod_cor	codice identificativo del corso	stringa	6 byte	obbligatorio
Titolo	titolo del corso	stringa	40 byte	obbligatorio
Docente	nome del docente del corso	stringa	20 byte	obbligatorio

Nome attributo	Descrizione	Tipo	Vincoli
Data	data in cui lo studente ha sostenuto l'esame	data	obbligatorio
Voto	voto dell'esame	numerico	obbligatorio

FISSA LE CONOSCENZE

1. Quando tra due entità esiste un'associazione di tipo N:N?
2. Quando si inseriscono attributi in un'associazione?

6. LE ASSOCIAZIONI BINARIE, UNARIE E MULTIPLE

Le associazioni possono essere **binarie** quando collegano due entità, **multiple** quando coinvolgono più di due entità e **unarie** quando associano elementi della stessa entità.

Associazioni binarie

Sono le più utilizzate, permettono di mettere in relazione due entità. Abbiamo già visto alcuni esempi di associazioni binarie descrivendo i tipi di relazioni.

Associazioni unarie (o ricorsive)

Si tratta di associazioni di **un'istanza di un'entità con un'altra istanza della stessa entità**. Per esempio, nell'entità Dipendente può esistere l'associazione Dirige, in cui un dipendente è a capo di altri dipendenti (fig. 18).

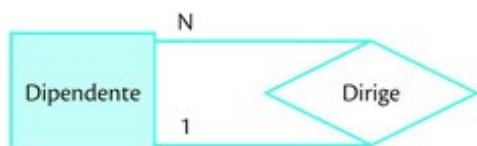


fig. 18 Associazione unaria

Associazioni multiple

Si tratta di associazioni in cui vengono coinvolte **più di due entità**, per esempio l'associazione tra Studente, Professore e Corso (fig. 19).

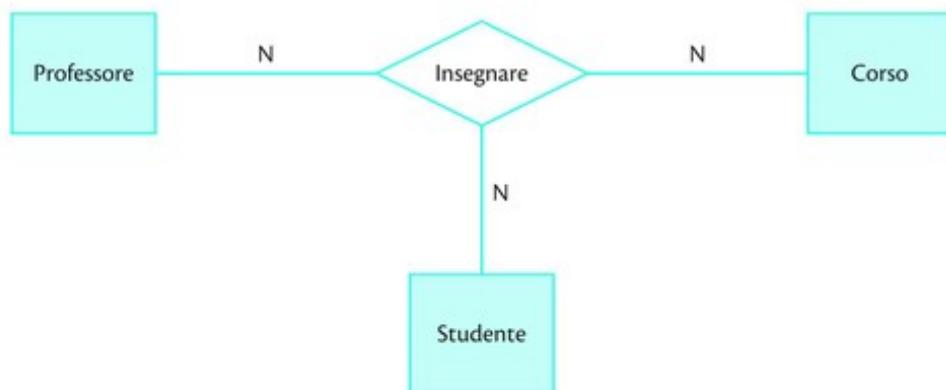


fig. 19 Associazione multipla

Tra due entità possono esistere anche più associazioni. Per esempio, tra le entità Persona e Casa possono esistere le associazioni Vende e Compra (fig. 20).

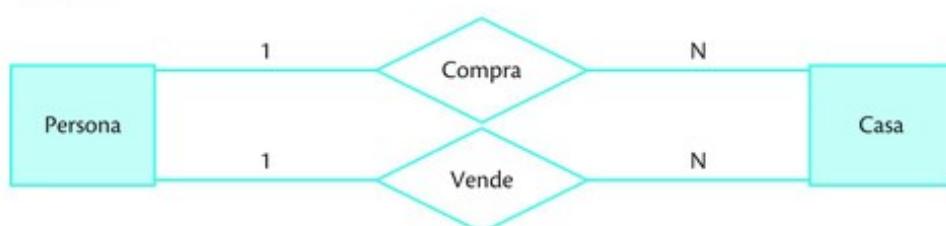


fig. 20 Associazioni diverse tra due entità

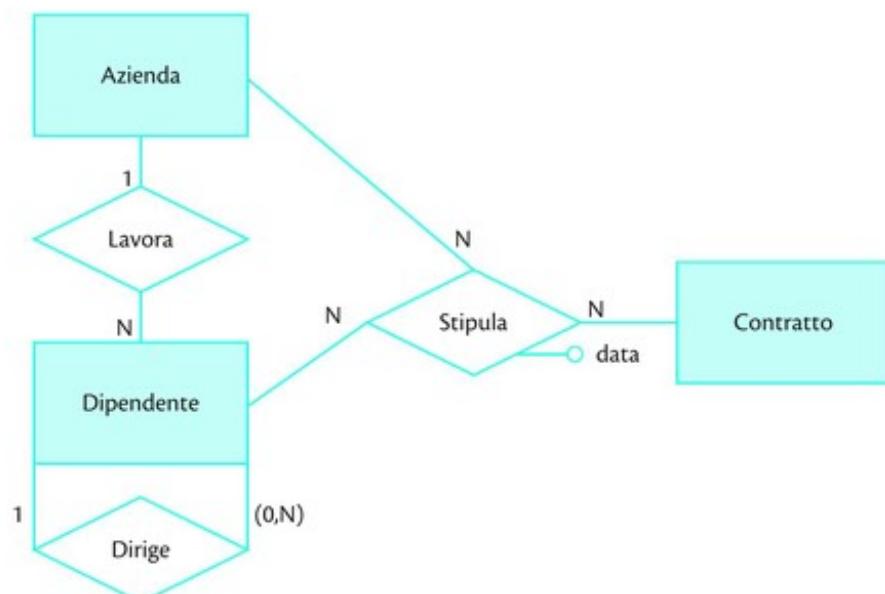
LABORATORIO

IL PROBLEMA Definire i tipi di associazioni possibili tra le seguenti entità: Dipendente, Azienda, Contratto.

L'ANALISI Tra le entità Dipendente, Azienda e Contratto possiamo identificare molteplici associazioni. Per esempio possiamo considerare:

- l'associazione binaria Lavora di tipo 1:N tra Azienda e Dipendente;
- l'associazione unaria Dirige per indicare che un Dipendente può essere a capo di altri Dipendenti (notare che non tutti i dipendenti sono dirigenti, per cui abbiamo inserito l'opzionalità nell'associazione);
- l'associazione multipla Stipula, che coinvolge le entità Dipendente, Azienda e Contratto, e rappresenta il tipo di contratto stipulato tra il Dipendente e l'Azienda. Nell'associazione possiamo inserire l'attributo data, per indicare la data in cui il contratto è stato firmato.

Lo schema E/R risultante sarà il seguente:



FISSA LE CONOSCENZE

1. Quali tipi di relazioni sono presenti in uno schema E/R?
2. In quale circostanza si parla di associazioni unarie?

7. ENTITÀ DEBOLI CON IDENTIFICAZIONE ESTERNA. GERARCHIE

■ Identificazione esterna

Nella definizione delle chiavi capita talvolta che l'entità presa in esame non abbia una sua chiave. In questi casi si ricorre all'**identificazione esterna**, si usano cioè attributi presi da altre entità con cui quella interessata è collegata. Tali entità si chiamano **entità deboli**.

In generale si dice che un'identificazione esterna avviene quando un'entità è individuata da una o più entità diverse, eventualmente con l'aggiunta di un attributo. Nella [fig. 21](#) vediamo alcuni esempi. Nel primo caso per comodità si è deciso, durante la fase di progettazione, di considerare un'entità Nave con alcune informazioni essenziali e un'entità Descrizione nave con maggiori dettagli. Tramite Nave viene identificata univocamente la Descrizione. Un altro esempio è quello di un Contratto che non può esistere se non viene identificato con l'Acquirente e il Venditore.

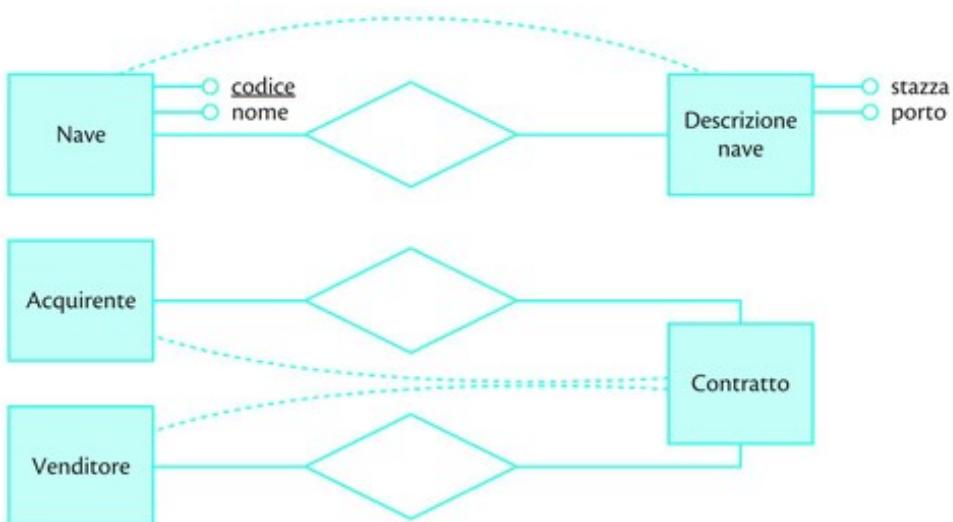


fig. 21 Esempi di identificazione esterna

Gerarchie

Le **gerarchie** sono costituite da insiemi di entità tra cui esistono associazioni di tipo gerarchico, tali che gli elementi delle entità a livello inferiore sono un sottoinsieme degli elementi dell'entità a livello superiore.

Le entità a livello inferiore ereditano tutte le proprietà e le associazioni dell'entità a livello superiore (non viceversa).

Una gerarchia si dice **completa**, se per ogni elemento dell'entità a livello superiore esiste almeno un elemento corrispondente di una delle entità a livello inferiore; se ciò non si verifica, si dice **non completa**.

Una gerarchia si dice **esclusiva**, se per ogni elemento dell'entità a livello superiore esiste al più un elemento corrispondente di una delle entità a livello inferiore.

In pratica si è di fronte a una gerarchia quando è possibile riconoscere e/o stabilire che un certo insieme X di entità costituisce un sottoinsieme di una classe di entità Y più ampia che le comprende.

Una gerarchia esclusiva e completa prende il nome di **gerarchia di generalizzazione** o **gerarchia ISA** (dall'inglese *is a*).

Un esempio di gerarchia di generalizzazione è quella dei Dipendenti di una scuola (superclasse Dipendente) in cui vi sono due sottoclassi (Docente e Personale ATA). Un dipendente farà parte sicuramente di una delle due sottoclassi (e solo di una), vi saranno degli attributi comuni (fig. 22) mentre alcuni attributi sono presenti solo in una sottoclasse.



fig. 22 Attributi comuni

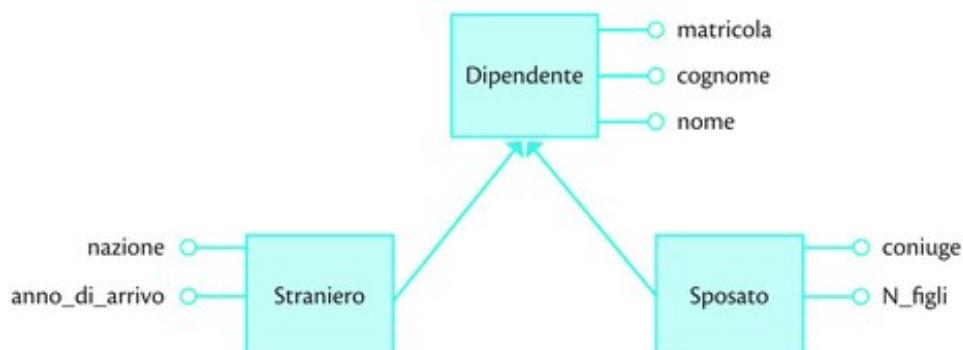
fig. 23 Gerarchia di generalizzazione

Un altro esempio di gerarchia di generalizzazione è mostrato nella fig. 23, dove si può notare che gli insiemi Uomo e Donna sono disgiunti.

Si chiamano **gerarchie di subset** quelle non esclusive e non complete: la sottoclasse individua un sottoinsieme non esclusivo di elementi.

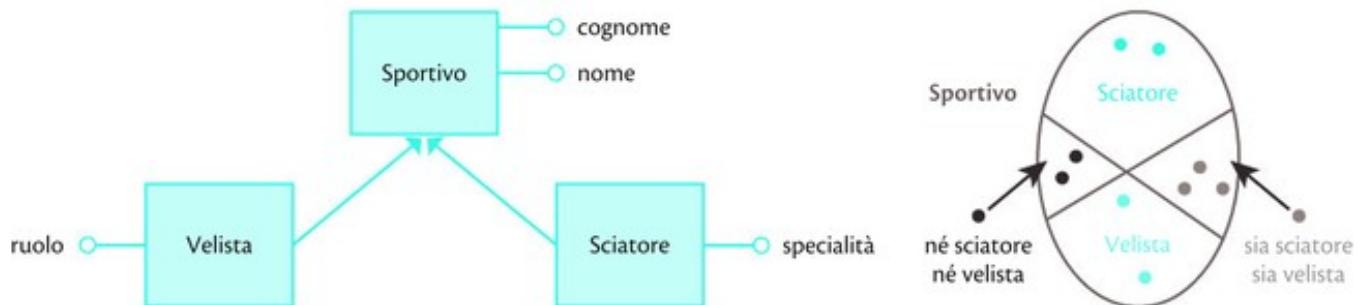
Per esempio (fig. 23), alla classe Dipendente associamo due sottoclassi in subset quali Straniero e Sposato. Ci potranno essere dipendenti che appartengono a tutte e due le sottoclassi (stranieri sposati), a una sola (stranieri oppure solo sposati) o a nessuna (celibi e non stranieri).

fig. 24 Subset



Un ulteriore esempio di subset è rappresentato nella **fig. 25**, dove viene mostrata la situazione anche attraverso gli insiemi: gli sciatori sono un sottoinsieme degli sportivi, come anche i velisti; però esistono anche sportivi che non praticano né lo sci né la vela o sportivi che praticano entrambi gli sport.

fig. 25 Subset



Nelle **fig. 26 a) e b)** sono mostrati degli esempi di gerarchie non corrette. Il primo esempio non è corretto, perché nome è un attributo presente in tutte le sottoentità; quindi, facendo parte degli attributi comuni, va messo nell'entità padre. Al contrario nel secondo esempio abbiamo un attributo opzionale nell'entità padre (anno di arrivo) che è significativo solo per una delle sottoentità.

a)

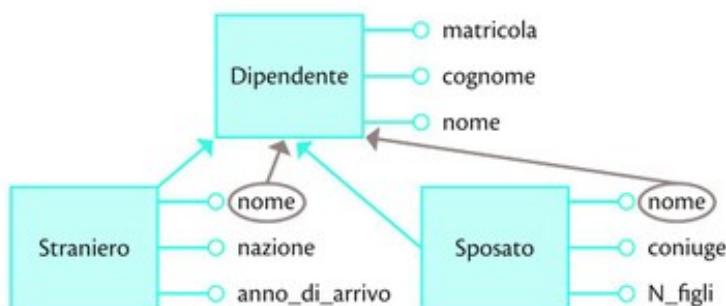
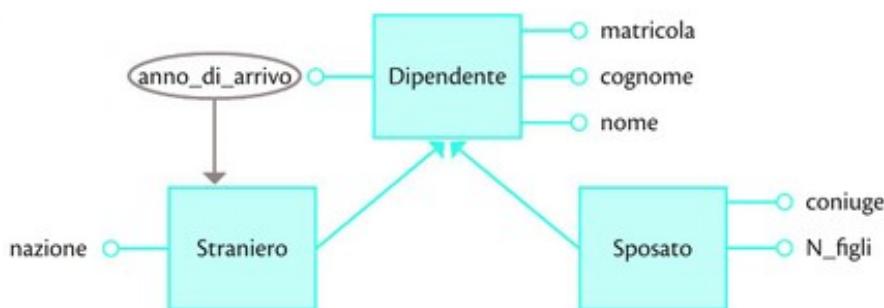


fig. 26 Esempi di gerarchie non corrette

b)



FISSA LE CONOSCENZE

1. In quali circostanze si parla di identificazione esterna?
2. Che cosa indica una gerarchia?
3. Che cos'è una gerarchia di generalizzazione?
4. Che cos'è una gerarchia di subset?

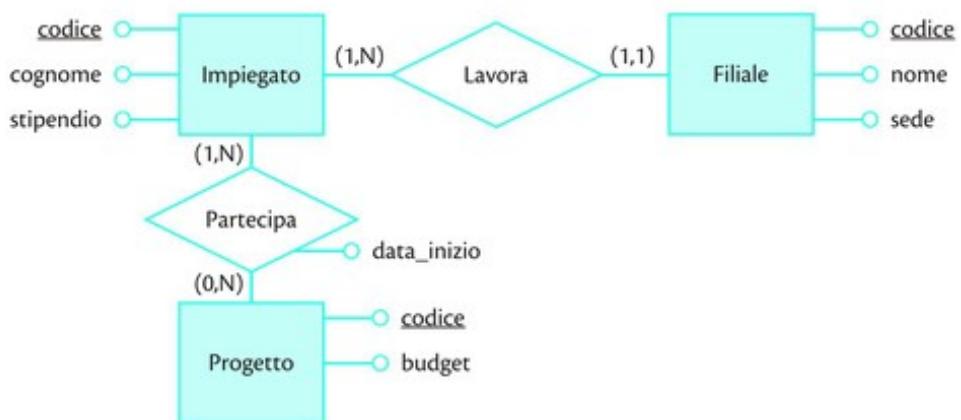
8. SCHEMI E SOTTOSCHEMI

L'insieme delle entità, delle relazioni e degli attributi individuati per rappresentare quella parte di realtà che interessa tutte le applicazioni determina lo **schema**.

La [fig. 27](#) mostra lo schema di una base di dati descritta con il modello E/R. In essa si evidenziano tre tipi di entità: Impiegato, Filiale e Progetto, ciascuna caratterizzata dai propri attributi.

Sono inoltre presenti due associazioni, Lavora e Partecipa, che permettono di realizzare i collegamenti tra le entità.

fig. 27 Esempio di schema E/R



Poiché un impiegato può occuparsi di più progetti e allo stesso modo a un progetto possono partecipare più impiegati, la relazione Partecipa tra Impiegato e Progetto è di tipo N:N; l'attributo della relazione (data_inizio) permette di conoscere la data di inizio attività degli impiegati rispetto ai diversi progetti.

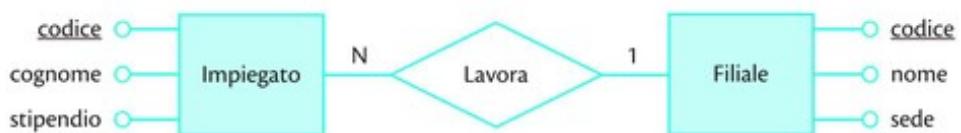
La relazione Lavora tra Filiale e Impiegato è di tipo 1:N, poiché presso una filiale possono lavorare più impiegati, ma un impiegato lavora presso una sola filiale della ditta.

La sua presenza permette di conoscere presso quale filiale lavorano gli impiegati.

Per **sottoschema** o **vista** (view) si intende una visione particolare dei dati relativa alla singola applicazione. Il sottoschema deriva dallo schema globale e semplifica la visione dei dati dei singoli utenti.

Se, per esempio, si vogliono conoscere quali impiegati lavorano in una filiale, si può definire un sottoschema dallo schema della [fig. 27](#) che veda solo le entità Impiegato e Filiale e la relazione Lavora, come illustra la [fig. 28](#).

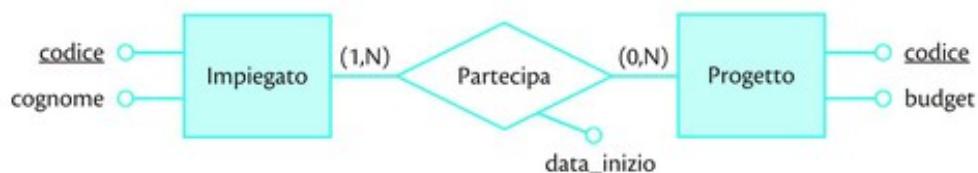
fig. 28 Esempio di vista



Su uno schema possono essere definiti diversi sottoschemi, ognuno dei quali descrive la **porzione di dati utilizzata dall'utente**, che è così autorizzato a interagire con la base di dati solo per quei dati sui quali ha la competenza.

In questo modo si eliminano errori per modifiche e cancellazioni inappropriate da parte di utenti non autorizzati e si garantisce che dati sensibili, come per esempio lo stipendio di un impiegato, siano visti solo dagli utenti consentiti.

L'applicazione che si occupa degli stipendi del personale sarà ovviamente interessata solo alla parte dello schema che contiene l'entità Impiegato (fig. 29), mentre quella riguardante la partecipazione degli impiegati ai vari progetti utilizzerà le entità Impiegato (della quale non vede l'attributo stipendio) e Progetto e la relazione Partecipa, come illustra la fig. 30.



Gli esempi visti finora sono molto semplici, perché il sottoschema consiste, in sostanza, di un sottoinsieme dello schema di partenza. In realtà le possibilità offerte da alcuni sistemi per la definizione di sottoschemi sono molto più vaste.

È possibile, per esempio, definire nuovi tipi di entità, utilizzando attributi delle entità esistenti: dallo schema della fig. 27 è possibile creare il sottoschema della fig. 31, nel quale è definita la nuova entità Sedi, contenente le informazioni degli impiegati e della filiale in cui lavorano i dipendenti.

Grazie al fatto che ogni applicazione è legata, solitamente, non a tutto lo schema, ma a un particolare sottoschema o vista, si può parlare di **indipendenza logica dei dati**. Essa è la caratteristica del DBMS che permette di modificare una parte dello schema logico dei dati, senza che si debbano rivedere le applicazioni che non fanno riferimento alla parte dello schema modificato.

Nel caso esaminato, volendo aggiungere l'attributo data_assunzione all'entità Impiegati, le applicazioni che lavorano con i sottoschemi della fig. 30 e della fig. 31 non sono influenzate dal cambiamento.

Ragionevolmente si può ipotizzare che la data di assunzione sia elaborata dall'applicazione che calcola gli stipendi degli impiegati: in questo caso il sottoschema della fig. 29 dovrà essere modificato e, di conseguenza, anche i programmi applicativi che interfacciano quel sottoschema.

FISSA LE CONOSCENZE

1. Che differenza c'è tra schema e sottoschema?
2. Perché è utile lavorare sul sottoschema?



fig. 29 Esempio di sottoschema

fig. 30 Esempio di sottoschema

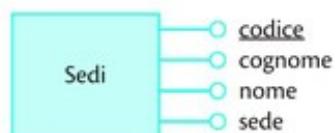


fig. 31 Esempio di sottoschema

9. PROGETTARE UN DATABASE

Nelle lezioni precedenti abbiamo visto come, attraverso il modello E/R, sia possibile usare i dati per rappresentare la realtà considerata attraverso entità, attributi e associazioni. Vogliamo ora mettere in pratica la metodologia descritta e in questa Lezione mostriamo come analizzare i dati e come effettuare l'analisi di un problema per definire lo schema concettuale della base di dati.

LABORATORIO

IL PROBLEMA Considera la struttura degli istituti bancari organizzati in agenzie, in cui sono depositati i conti correnti dei clienti. Per ogni agenzia devi ottenere informazioni relative al direttore (nome, cognome, mail). La base di dati deve memorizzare i dati relativi ai movimenti effettuati su ciascun c/c (importo, tipo e data). Il tipo del movimento indica se si tratta di prelievo o di versamento. In ogni momento deve essere possibile conoscere il saldo del c/c.

L'ANALISI La rappresentazione dei dati deve tenere conto del fatto che esistono più istituti bancari, a ognuno dei quali sono associate più agenzie, con un rapporto gerarchico. In ogni agenzia sono depositati i c/c dei clienti a cui fanno riferimento i movimenti bancari. Lo schema E/R che rappresenta i dati del problema è illustrato nella seguente figura.

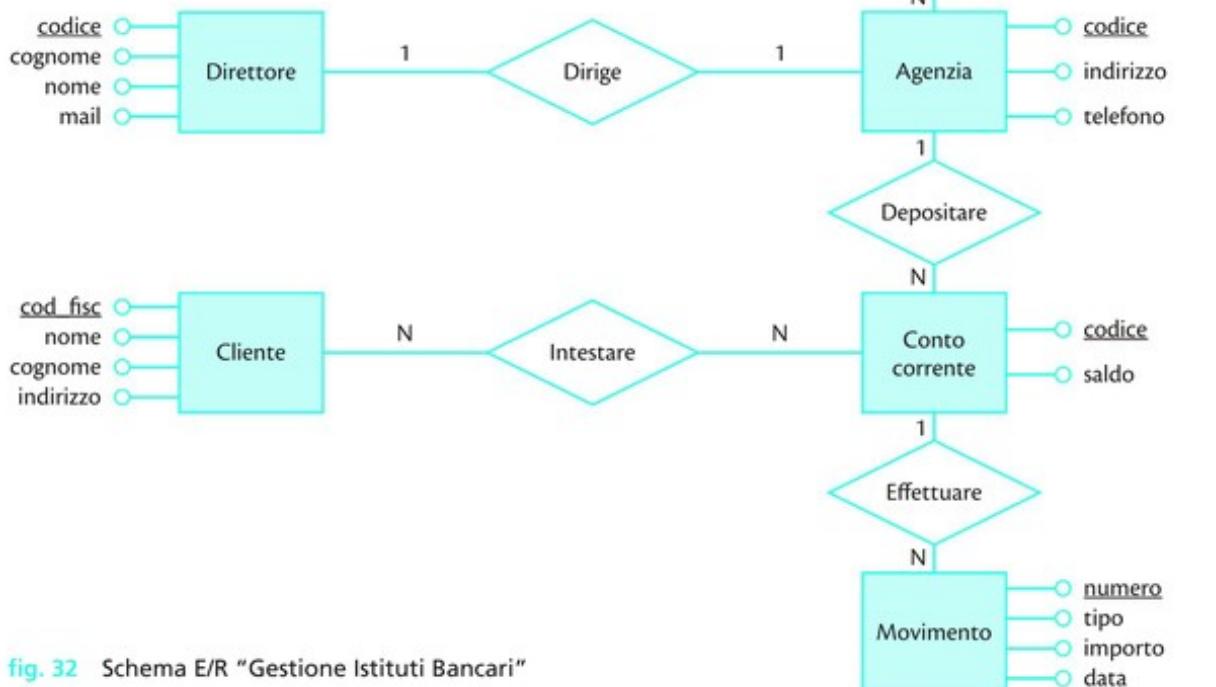


fig. 32 Schema E/R "Gestione Istituti Bancari"

DIZIONARIO ENTITÀ

Nome entità	Descrizione	Attributi	Identificatore
Banca	lista degli istituti bancari	codice denominazione sede	codice
Agenzia	agenzie di ciascun istituto bancario	codice indirizzo telefono	codice
Direttore	direttore di ciascuna agenzia	codice nome cognome mail	codice
Conto corrente	conti correnti di ciascuna agenzia	codice saldo	codice
Movimento	movimenti relativi a ciascun conto corrente	numero tipo importo data	numero
Cliente	clienti di ciascuna agenzia	cod_fisc nome cognome indirizzo	cod_fisc

DIZIONARIO ASSOCIAZIONI

Nome associazione	Descrizione	Entità coinvolte	Attributi
Appartiene	associa una banca a ciascuna agenzia	Banca (1,1) Agenzia (1,N)	
Dirige	associa una agenzia con il proprio direttore	Agenzia (1,1) Direttore (1,1)	
Depositare	associa un'agenzia con i suoi conti correnti	Agenzia (1,1) Conto corrente (1,N)	
Effettuare	associa un conto corrente con i movimenti effettuati su di esso	Conto corrente (1,1) Movimento (1,N)	
Intestare	associa un conto corrente ai suoi clienti intestatari	Conto corrente (1,N) Cliente (1,N)	

Tra l'entità Banca e l'entità Agenzia esiste la relazione uno a molti (Appartiene), poiché a un istituto bancario possono appartenere più agenzie, ma un'agenzia può appartenere a una sola banca (fig. 33).

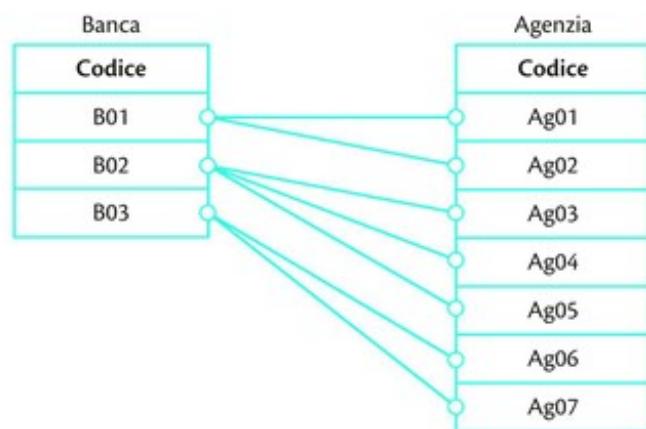


fig. 33 Associazione 1:N tra Banca e Agenzia

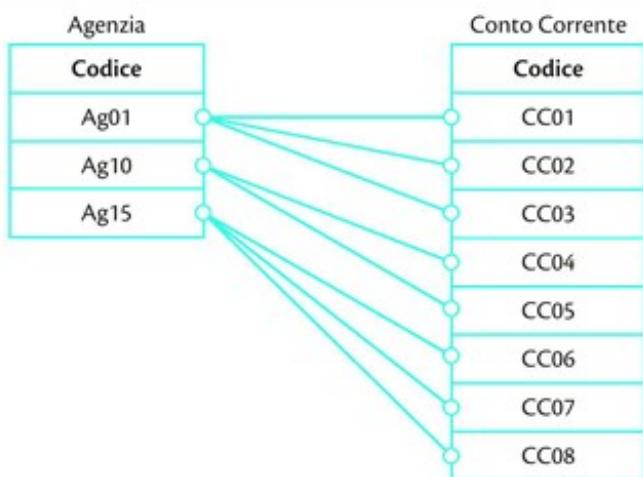
Tra l'entità Agenzia e l'entità Direttore esiste la relazione uno a uno (Dirige), poiché un'agenzia è diretta da un unico direttore e un direttore ha la responsabilità di una sola agenzia (fig. 34).

fig. 34 Associazione 1:1 tra Direttore e Agenzia

Direttore		Agenzia
Codice	Cognome	Codice
D025	Rossi	Ag01
D001	Verdi	Ag10
D020	Bruni	Ag15

Tra l'entità Agenzia e l'entità Conto corrente esiste la relazione uno a molti (Depositare), poiché in un'agenzia possono essere depositati molti conti correnti, ma un conto corrente può essere depositato in una sola agenzia (fig. 35).

fig. 35 Associazione 1:N tra Agenzia e Conto corrente



Tra l'entità Conto corrente e l'entità Movimento esiste la relazione uno a molti (Effettuare), poiché su un conto corrente possono essere effettuati più movimenti, ma un movimento si riferisce a un solo conto corrente (fig. 36).

fig. 36 Associazione 1:N tra Conto corrente e Movimento

Conto corrente		Movimento		
Codice		Numero	Tipo	Importo
CC01	1	1	P	100
CC02	2	2	V	200
CC03	3	3	V	140
	4	4	P	250
	5	5	P	500
	6	6	V	800
	7	7	P	50
	8	8	V	170

Tra l'entità Conto corrente e l'entità Cliente esiste la relazione molti a molti (Intestare), poiché un conto corrente può essere intestato a più clienti e un cliente può aprire più di un conto corrente (fig. 37).

fig. 37 Associazione N:N tra Cliente e Conto corrente

Cliente		Conto corrente
Codice	Cognome	Codice
CC01	Neri	CC01
CC02	Bianchi	CC02
CC03	Rossi	CC03
		CC04

DESCRIZIONE DEGLI ATTRIBUTI

Si possono riassumere le informazioni presenti nello schema E/R con le seguenti tabelle.

BANCA

Nome attributo	Descrizione	Tipo	Lunghezza	Vincoli
Codice	codice dell'istituto bancario	stringa	4 byte	obbligatorio
Denominazione	denominazione dell'istituto bancario	stringa	20 byte	obbligatorio
Sede	indirizzo dell'istituto bancario	stringa	30 byte	obbligatorio

AGENZIA

Nome attributo	Descrizione	Tipo	Lunghezza	Vincoli
Codice	codice agenzia	stringa	4 byte	obbligatorio
Indirizzo	indirizzo dell'agenzia	stringa	20 byte	obbligatorio
Telefono	numero di telefono dell'agenzia	stringa	30 byte	obbligatorio

DIRETTORE

Nome attributo	Descrizione	Tipo	Lunghezza	Vincoli
Codice	codice del direttore di agenzia	stringa	4 byte	obbligatorio
Nome	nome del direttore di agenzia	stringa	20 byte	obbligatorio
Cognome	cognome del direttore di agenzia	stringa	30 byte	obbligatorio
Mail	indirizzo e-mail del direttore di agenzia	stringa	30 byte	opzionale

CONTO CORRENTE

Nome attributo	Descrizione	Tipo	Lunghezza	Vincoli
Codice	codice del conto corrente	stringa	15 byte	obbligatorio
Saldo	saldo del conto corrente	numerico	8 byte	obbligatorio

MOVIMENTO

Nome attributo	Descrizione	Tipo	Lunghezza	Vincoli
Numero	numero identificativo del movimento	numerico	4 byte	obbligatorio
Tipo	tipo di movimento	carattere	1 byte	Valori 'P' o 'V'
Importo	importo del movimento	numerico	6 byte	obbligatorio
Data	data del movimento	data	8 byte	obbligatorio

CLIENTE

Nome attributo	Descrizione	Tipo	Lunghezza	Vincoli
Cod_fisc	codice fiscale del cliente	stringa	16 byte	obbligatorio
Nome	nome del cliente	stringa	20 byte	obbligatorio
Cognome	cognome del cliente	stringa	30 byte	obbligatorio
Indirizzo	indirizzo del cliente	stringa	20 byte	obbligatorio



RIPASSIAMO INSIEME

Progettazione di un database

La progettazione di una base di dati si articola in quattro fasi:

- **raccolta e analisi requisiti:** consiste nel reperire la documentazione e nell'analizzare la situazione esistente;
- **progettazione concettuale:** sono descritti in modo formale i dati visti dagli utenti;
- **progettazione logica:** trasforma lo schema concettuale prodotto nella fase precedente nello schema logico su cui si basa il DBMS scelto;
- **progettazione fisica:** vengono definite le strutture fisiche di memorizzazione con i relativi parametri.

Il modello E/R - Entità e attributi

Modellare i dati significa costruire una **rappresentazione semplificata** della realtà osservata, individuandone gli elementi caratterizzanti (entità) e i legami intercorrenti tra essi (relazioni). L'**entità** è un oggetto (concreto o astratto) che ha un significato quando viene considerato in modo isolato ed è di interesse per la realtà che si vuole modellare. Gli **attributi** sono le proprietà delle entità e sono caratterizzati dal tipo, dalla dimensione e dall'opzionalità (non necessariamente l'attributo deve avere un valore).

Le chiavi

Le **chiavi** sono uno o più attributi di un'entità che identificano **univocamente** le istanze dell'entità stessa. Tra le possibili chiavi candidate si sceglie quella che sarà la chiave primaria.

Le relazioni 1:1 e 1:N

La **relazione** è un legame che stabilisce un'interazione tra le entità. Una relazione

può essere di **tipo 1:1**, quando a ogni elemento della prima entità corrisponde un solo elemento della seconda entità e viceversa. Se l'associazione è di **tipo 1:N** significa che a ogni elemento della prima entità possono corrispondere più elementi della seconda entità, ma non viceversa.

Le associazioni N:N e le relazioni con attributi

Un'associazione di **tipo N:N** tra due entità indica che a ogni istanza della prima entità possono corrispondere più istanze della seconda entità e viceversa. Anche le associazioni, come le entità, possono avere degli attributi. Un attributo indica una caratteristica.

Le associazioni binarie, unarie e multiple

Le associazioni possono essere **unarie**, se mettono in relazione un'entità con se stessa; **binarie**, se mettono in relazione due entità, e **multiple**, se mettono in relazione un numero di entità maggiore di due.

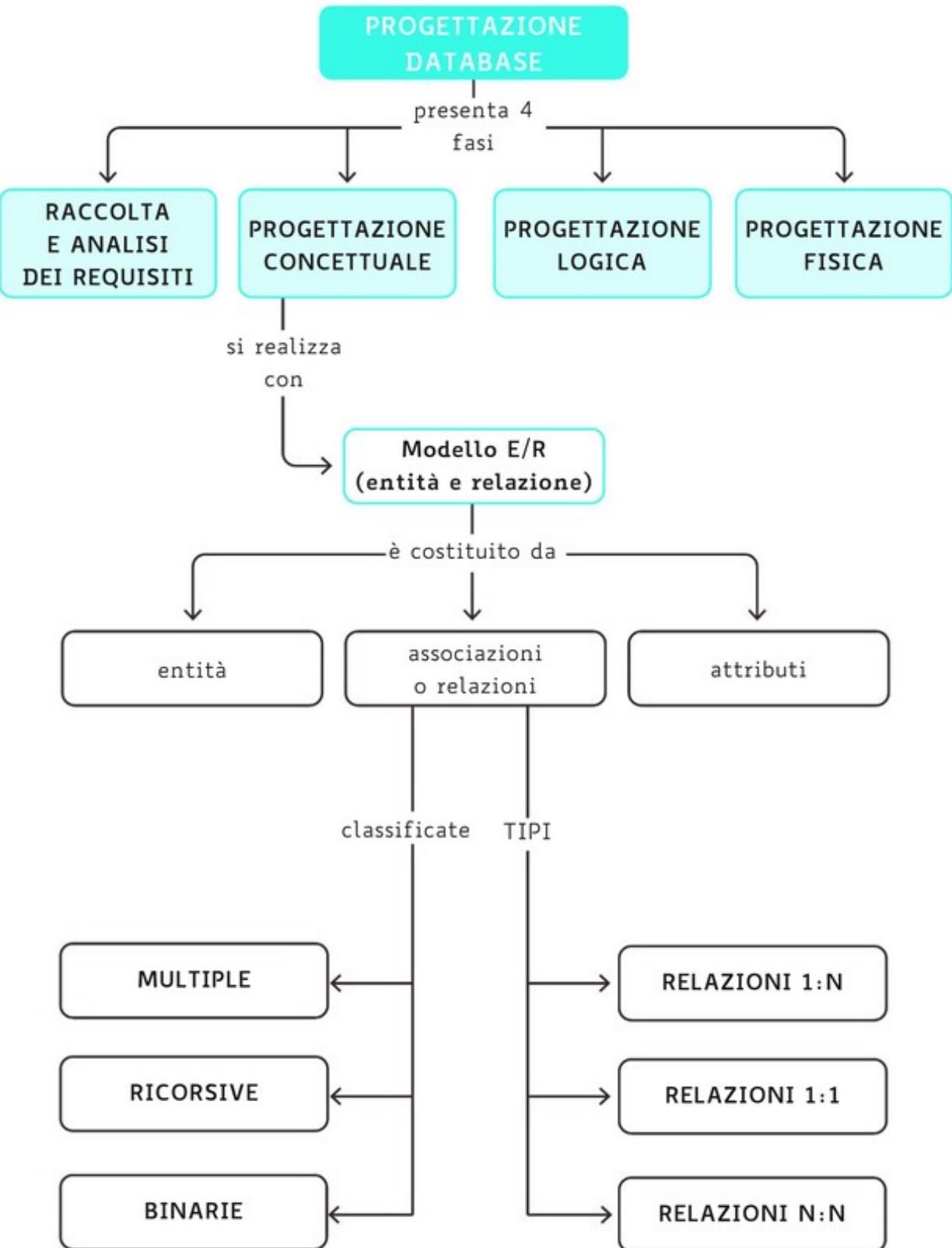
Entità deboli con identificazione esterna. Gerarchie

Si chiamano **entità deboli** quelle entità che necessitano di una **identificazione esterna**.

Si parla di **gerarchie** quando esistono delle associazioni tra entità formate da elementi che sono dei sottoinsiemi degli elementi di entità a livello superiore. Esistono gerarchie esclusive e complete (ISA) e gerarchie non esclusive e non complete (subset).

Schemi e sottoschemi

Per **sottoschema** o **vista** si intende una visione particolare dei dati relativa alla singola applicazione.



TEST**TEST**

Svolgi il test interattivo

Vero o falso?

1. L'associazione ricorsiva è tra due entità. V F
2. Qualsiasi attributo di un'entità può essere chiave primaria. V F
3. Lo schema E/R è la rappresentazione dello schema concettuale. V F
4. Una gerarchia è una tipologia di associazione tra entità. V F
5. Un'associazione non può avere attributi. V F

Scelta multipla (una sola è la risposta esatta)

6. Il modello E/R descrive:
 - A lo schema fisico di una base di dati.
 - B una singola vista.
 - C lo schema concettuale di una base di dati.
 - D i dati di interesse.
7. Gli attributi esprimono le caratteristiche:
 - A delle chiavi primarie.
 - B delle relazioni.
 - C di entità e relazioni.
 - D di entità e chiavi candidate.
8. Una gerarchia ISA è:
 - A completa e non esclusiva.
 - B esclusiva e non completa.
 - C non completa e non esclusiva.
 - D completa ed esclusiva.
9. Una chiave candidata è:
 - A una possibile chiave primaria.
 - B può avere dei duplicati.
 - C non identifica le istanze di una entità in modo univoco.
 - D è formata da un solo attributo.
10. Una relazione 1:N tra l'entità A e l'entità B indica che:
 - A a una istanza di A corrispondono più istanze di B.
 - B a una istanza di A corrispondono più istanze di B, mentre a una istanza di B corrisponde una e una sola di A.
 - C a più istanze di B corrisponde una e una sola di A.
 - D a una istanza di A corrispondono più istanze di B, ma non viceversa.

PREPARATI PER IL COLLOQUIO ORALE

1. Che cosa rappresenta il modello E/R? [lez. 2]
2. Che cosa indica una chiave primaria? [lez. 3]
3. Come si indica la cardinalità minima di un'associazione? [lez. 4]
4. Che cos'è una gerarchia? [lez. 7]
5. Quando un attributo si riferisce ad una relazione? [lez. 5]
6. Che cos'è una vista? [lez. 8]
7. Elencare i tipi di associazioni possibili tra due entità. [lez. 4 - lez. 5]
8. Che caratteristiche ha una gerarchia di subset? [lez. 7]

CLIL – IN ENGLISH, PLEASE



AUDIO
Ascolta la pronuncia
del testo

ABSTRACT

The Entity/Relationship Model

The Entity/Relationship Model or Diagram (ERD) describes, at a conceptual level, the entities (the objects of interest to the database) and their relationships (the links connecting them).

The ER diagram is an abstract data model: it's a data structure which can be implemented in a database, typically a relational database.

Relationships have different cardinalities (1:1, 1:N, N:N), can be optional and may have attributes like entities do.

In an Entity/Relationship Diagram of a data model, one or more unique keys may be declared for each data entity. Each unique key is composed from one or more data attributes of that data entity.

The set of unique keys declared for a data entity is often referred to as the candidate keys for that data entity. From the set of candidate keys, a single unique key is selected and declared the primary key for that data entity.

The most important criteria for consideration of a primary key are:

- uniqueness;
- simplicity (so that relational representation can be simpler);
- stability (should not be altered frequently);
- familiarity (meaningful to the user).

EXERCISES

True or false?

1. The ERD represents the conceptual schema. T F
2. Any attribute of an entity may be a primary key. T F
3. A relationship is an object of interest to the database. T F
4. The most common databases are relational. T F
5. A relationship can never have attributes. T F
6. A primary key cannot be altered whatsoever. T F

Multiple choice

7. Attributes specify characteristics of:
 - A primary keys.
 - B relationships.
 - C entities and relationships.
 - D entities and candidate keys.
8. ERDs describe:
 - A the physical schema of a data set.
 - B a single view.
 - C the conceptual schema of a data set.
 - D interesting data.

GLOSSARY

Entity: any object in the system that we have to model. In general entities are recognizable concepts such as person, places, things, or events which have relevance to the database.

Relationship: represent a link between two or more entities.

Attribute: a property inherent in a database entity or associated with that entity for database purposes.

Candidate key: a set of data that can uniquely identify any database record without referring to any other data.

Primary key: a choice of candidate key; any other candidate key is called alternate key.



GLOSSARIO
CLIL

**COMPETENZE DISCIPLINARI**

- Identificare e applicare le metodologie e le tecniche della gestione per progetti.
- Redigere relazioni tecniche e documentare le attività individuali e di gruppo relative a situazioni professionali.
- Interpretare i sistemi aziendali nei loro modelli, processi e flussi informativi con riferimento alle differenti tipologie di imprese.

COMPETENZE DEL XXI SECOLO**Abilità fondamentali****CULTURA SCIENTIFICA**

Capacità di usare conoscenze scientifiche e regole/modelli per interpretare un fenomeno.

Competenze trasversali**PENSIERO CRITICO**

Saper analizzare e valutare situazioni in modo da impiegare informazioni e idee per formulare risposte e soluzioni.

COMUNICAZIONE

Saper ascoltare, comprendere e contestualizzare le informazioni, per poi trasmetterle ad altri (in modalità verbale o non verbale).

COLLABORAZIONE

Saper lavorare in gruppo in vista di un obiettivo comune, prevenendo ed eventualmente gestendo i conflitti.

Qualità caratteriali**CURIOSITÀ**

Inclinazione a porre domande con una mentalità aperta.

**OBIETTIVI FORMATIVI**

- Analizzare una situazione aziendale e individuare i dati.
- Realizzare un diagramma Entità-Relazioni e documentarlo opportunamente.
- Consultare fonti Internet.
- Esporre i risultati della ricerca alla classe.

TEMPI

- Analisi e soluzione del problema d'esame: 3 ore.
- Preparazione della ricerca e del PowerPoint: 2 ore.
- Presentazione dei risultati e dibattito in classe: 1 ora.
- Autovalutazione: 10 minuti.

STRUMENTI

- Libro di testo.
- Dispositivo connesso a Internet.
- Software PowerPoint.
- Proiettore collegato al computer in classe o in laboratorio.

IL TEMA

Di seguito è riportato un estratto della prima parte della **prova di informatica** dell'Esame di Stato del 2015 per l'Istituto Tecnico, settore Tecnologico, indirizzo Informatica e Telecomunicazioni.

Da solo o in gruppo leggi il tema proposto e svolgilo. In questa unità prendiamo in considerazione i punti 1 e 2.

LA WEB COMMUNITY DEGLI EVENTI LIVE

Si vuole realizzare una web community per condividere dati e commenti relativi a eventi dal vivo di diverse categorie, ad esempio concerti, spettacoli teatrali, balletti, ecc. che si svolgono in Italia.

Gli eventi vengono inseriti sul sistema direttamente dai membri stessi della community, che si registrano sul sito fornendo un nickname, nome, cognome, indirizzo email e scegliendo una o più categorie di eventi a cui sono interessati.

Ogni membro iscritto riceve periodicamente per posta elettronica una newsletter, emessa automaticamente dal sistema, che riporta gli eventi delle categorie da lui scelte, che si svolgeranno nella settimana seguente nel territorio provinciale dell'utente.

I membri registrati possono interagire con la community sia inserendo i dati di un nuovo evento, per il quale occorre specificare categoria, luogo di svolgimento, data, titolo dell'evento e artisti coinvolti, sia scrivendo un post con un commento e un voto (da 1 a 5) su un evento. Il sito della community offre a tutti, sia membri registrati sia utenti anonimi, la consultazione dei dati online, tra cui:

- visualizzazione degli eventi di un certo tipo in ordine cronologico, con possibilità di filtro per territorio di una specifica provincia;
- visualizzazione di tutti i commenti e voti relativi ad un evento.

Il candidato, fatte le opportune ipotesi aggiuntive, sviluppi:

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.*
- [...]

I nuclei tematici fondamentali e gli obiettivi

Fra i nuclei tematici fondamentali e gli obiettivi cui si riferiscono le prove d'esame (vedi la sezione finale "La preparazione alla seconda prova scritta dell'esame di Stato"), questo estratto si riferisce ai seguenti.

Nuclei tematici fondamentali	Obiettivi
<ul style="list-style-type: none"> • Progettazione di basi di dati: modellazione concettuale. 	<ul style="list-style-type: none"> • Affrontare situazioni problematiche, utilizzando adeguate strategie cognitive e procedure operative orientate alla progettazione di soluzioni informatiche. • Realizzare progetti secondo procedure consolidate e criteri di sicurezza.

Svolgimento

Analisi della realtà e progetto concettuale della base di dati

Analizzando il testo, ci rendiamo conto che i dati da gestire sono relativi a:

- gli **Eventi** che si vogliono registrare e che si possono suddividere in **Categorie**;
- gli **Utenti** che inseriscono, consultano e commentano gli eventi stessi;
- gli **Artisti**, entità collegata con eventi, che introduciamo per rappresentare i dati degli artisti che partecipano agli eventi,

Le entità da rappresentare potrebbero quindi essere:

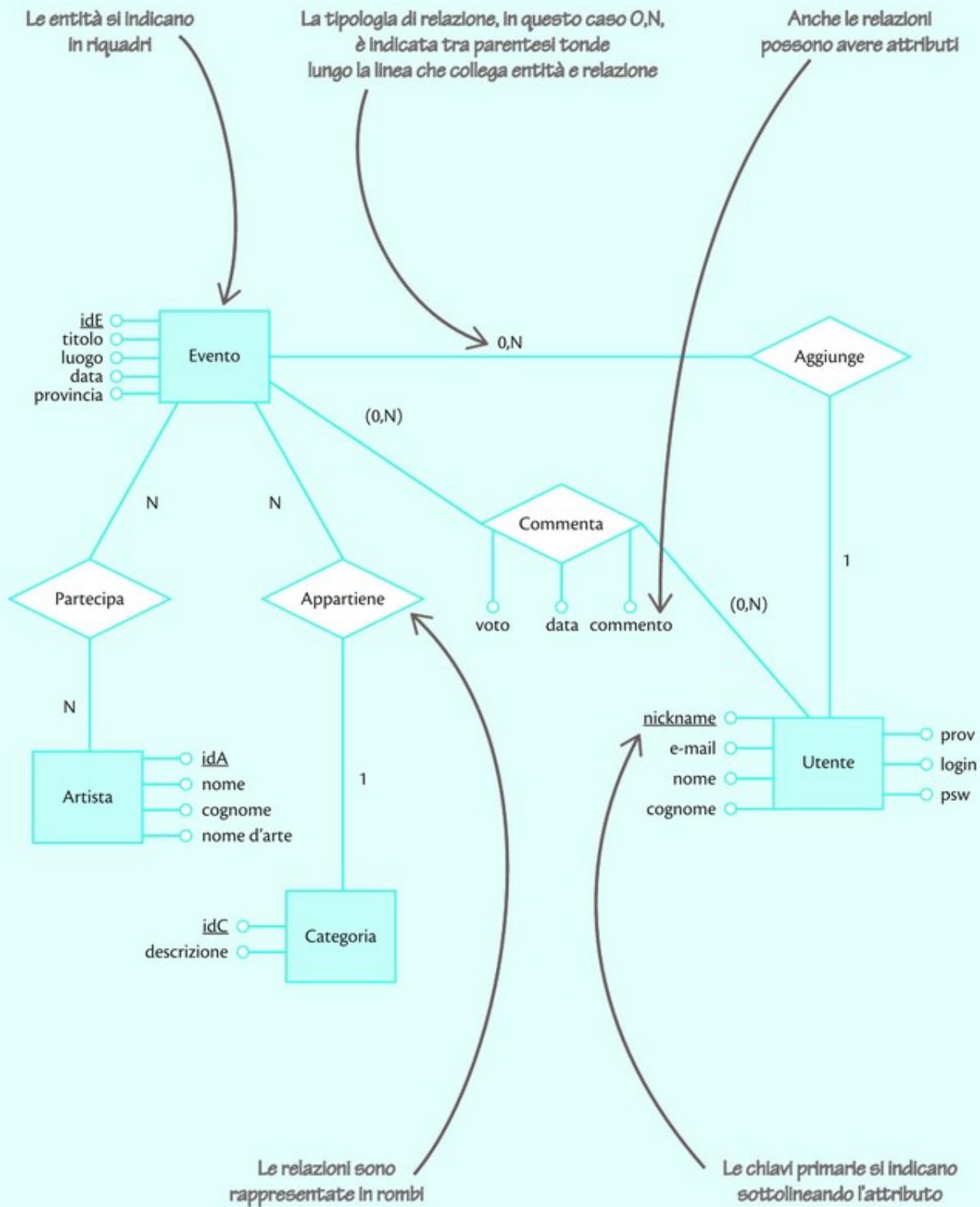
- Evento;
- Artista;
- CATEGORIA;
- Utente.

Nel testo si parla di due **tipologie di utenti**, gli utenti occasionali e quelli registrati, per cui si potrebbe pensare di inserire una **gerarchia ISA** di utente che abbia, come sottoentità, Utente occasionale e Utente registrato.

Analizzando meglio il problema, però, ci si rende conto che non ha senso memorizzare i dati degli utenti occasionali, poiché per loro non si ha nessun dato a disposizione.

È importante aggiungere l'**attributo provincia** sia per l'Evento (provincia di svolgimento) sia per l'Utente (provincia di residenza), dati di cui il sistema ha bisogno per inviare a ciascun utente della *community* le newsletter con gli eventi che si svolgono nella sua provincia di residenza.

Un possibile schema E/R per rappresentare il problema è il seguente.



DIZIONARIO ENTITÀ

Nome entità	Descrizione	Attributi	Identificatore
Evento	Lista degli eventi (concerti, spettacoli, balletti, ecc.) inseriti nella base dati	idE titolo luogo data provincia	idE
Artista	Artisti coinvolti negli eventi considerati	idA nome cognome nome d'arte	idA
Categoria	Tipologia a cui può appartenere un evento	idC descrizione	idC
Utente	Dati che un utente inserisce al momento della registrazione	nickname e-mail nome cognome prov login psw	nickname

DIZIONARIO ASSOCIAZIONI

Nome associazione	Descrizione	Entità coinvolte	Attributi
Preferisce	Indica le categorie preferite da un utente	Utente (1,N) Categoria (1,N)	
Partecipa	Indica gli artisti che partecipano a un evento	Evento (1,N) Artista (1,N)	
Appartiene	Associa le categorie agli eventi	Categorie (1,1) Evento (1,N)	
Commenta	Rappresenta i commenti che gli utenti possono inserire sugli eventi	Utente (1,N) Evento (1,N)	voto data commento
Aggiunge	Indica l'utente che ha inserito l'evento	Utente (1,1) Evento (1,N)	

- Tra l'entità Evento e l'entità Artista esiste una **relazione di tipo N:N**, poiché a un evento possono partecipare uno o più artisti e un artista può partecipare a uno o più eventi.
- Tra Evento e Categoria è stata inserita un'**associazione di tipo 1:N**, perché si suppone che un evento appartenga a una sola categoria, mentre a una categoria possono appartenere più eventi.
- Come evidenziato nel testo, un utente ha preferenze per più categorie, per cui la relazione Preferisce introdotta tra Utente e Categoria è di **tipo N:N**; infatti un utente sceglie più categorie e una categoria può essere scelta da più utenti.
- La relazione Commenta presente tra Utente ed Evento ha **tre attributi** (data, voto, commento) ed è di **tipo N:N**, poiché un utente può esprimere zero o più commenti e a un evento possono essere associati più commenti.
- Tra Utente ed Evento abbiamo inserito la relazione Aggiunge, che indica la possibilità da parte di un utente di aggiungere zero o più eventi che si svolgono nella sua provincia di appartenenza. L'associazione sarà di **tipo 1:N**, poiché un evento viene aggiunto da un solo utente.

DIZIONARIO ATTRIBUTI**EVENTO**

Nome	Descrizione	Tipo	Vincoli
idE	Identificativo dell'evento	stringa	Non più di 8 caratteri. Non nullo
titolo	Titolo dell'evento	stringa	
data	Data in cui si svolge l'evento	data	
luogo	Città in cui si svolge l'evento	stringa	
provincia	Provincia in cui si svolge l'evento	stringa	

UTENTE

Nome	Descrizione	Tipo	Vincoli
nickname	Identificativo dell'utente	stringa	Non più di 12 caratteri. Non nullo
e-mail	Indirizzo email dell'utente	stringa	Deve contenere il carattere @
nome	Nome dell'utente	stringa	
cognome	Cognome dell'utente	stringa	
prov	Provincia di residenza dell'utente	stringa	
login	Username scelto dall'utente per accedere al sistema	stringa	
psw	Password scelta dall'utente per accedere al sistema	stringa	

ARTISTA

Nome	Descrizione	Tipo	Vincoli
idA	Identificativo dell'artista	stringa	Non più di 10 caratteri. Non nullo
nome	Nome dell'artista	stringa	
cognome	Cognome dell'artista	stringa	
nome d'arte	Nome d'arte dell'artista	stringa	

CATEGORIA

Nome	Descrizione	Tipo	Vincoli
idC	Identificativo della categoria	stringa	Non più di 8 caratteri. Non nullo
descrizione	Breve descrizione della categoria	stringa	

COMMENTA

Nome	Descrizione	Tipo	Vincoli
voto	Voto attribuito da un utente a un evento	intero	Valori compresi da 1 a 5 estremi inclusi
data	Data di attribuzione del voto	data	
commento	Commento sull'evento	stringa	Non più di 60 caratteri

COMPITI DI REALTÀ

Dopo aver confrontato la risoluzione del tema d'esame con la tua, svolgi in gruppo alcune delle seguenti attività.

- 1. COMPETENZA DIGITALE** **COLLABORAZIONE** Effettuate una ricerca su Internet e analizzate lo **stato dell'arte** delle web *community* e dei *social network*: potete partire da questo link <https://tinyurl.com/y9vy-6svd>, in cui la piattaforma *We Are Social* fornisce un'analisi di tipo quantitativo, e da questo articolo di SkyTG24 <https://tinyurl.com/ydeb389p>, che individua alcune delle più importanti problematiche connesse con l'uso dei *social*. Potete usare anche altre fonti più recenti, ma ricordatevi di verificare sempre l'**attendibilità** dei siti web che consultate.
- 2. PENSIERO CRITICO** **COLLABORAZIONE** Individuate quella che, secondo voi, è la **problematica più attuale** collegata all'uso dei *social network*: può essere la *privacy*, il tema delle *fake news*, o altri argomenti che vi sembrano sentiti e importanti. Approfonditeli, discutendo in gruppo.
- 3. COMPETENZA DIGITALE** **COLLABORAZIONE** Raccogliete i **risultati della vostra ricerca** e della vostra discussione in una **presentazione in PowerPoint** (o in un altro software di presentazione), formata al massimo da cinque slide.
- 4. COLLABORAZIONE** **COMUNICAZIONE** In classe, con la supervisione dell'insegnante, condividete la presentazione con i compagni: confrontate le **tematiche emerse** e discutetele.
- 5. CULTURA SCIENTIFICA** **COLLABORAZIONE** Infine, in classe, discutete con i compagni sullo *schema E/R* presentato nella risoluzione al tema d'esame e proponete **eventuali modifiche** che vi sembrano adatte a rappresentare, per **completezza e realismo**, la costruzione del database richiesto all'esame di Stato.

AUTOVALUTAZIONE

Al termine delle attività rifletti sull'esperienza e completa la tabella di autovalutazione.

Attività	LIVELLO		
	Base	Intermedio	Avanzato
Mi sono orientato correttamente nella risoluzione del tema proposto all'esame di Stato?	Ho seguito la risoluzione proposta, ma non sarei in grado di progettare il database in autonomia, se non in modo poco dettagliato o sommario. <input type="checkbox"/>	Ho seguito la risoluzione proposta, e sarei in grado di progettare il database da solo, anche se avrei tralasciato alcuni particolari. <input type="checkbox"/>	Ho seguito la risoluzione proposta e sarei in grado di realizzare uno schema concettuale completo e ben descritto. <input type="checkbox"/>
Ho interagito con i membri del mio gruppo in modo proficuo e costruttivo?	Ho lavorato principalmente da solo. <input type="checkbox"/>	Ho collaborato con i compagni in tutte le attività. <input type="checkbox"/>	Ho collaborato con i compagni e in alcuni casi ho guidato il lavoro del gruppo. <input type="checkbox"/>
Ho partecipato attivamente al dibattito in classe?	Non sono riuscito a collaborare alle attività del mio gruppo. <input type="checkbox"/>	Ho partecipato al dibattito ma il mio contributo è stato marginale. <input type="checkbox"/>	Ho partecipato al dibattito e il mio contributo è stato determinante. <input type="checkbox"/>
Sono in grado di usare gli strumenti informatici?	Ho alcune difficoltà nell'uso di PowerPoint. <input type="checkbox"/>	Uso correttamente PowerPoint ma in modo amatoriale. <input type="checkbox"/>	Uso PowerPoint da utente esperto. <input type="checkbox"/>

3

Modello relazionale



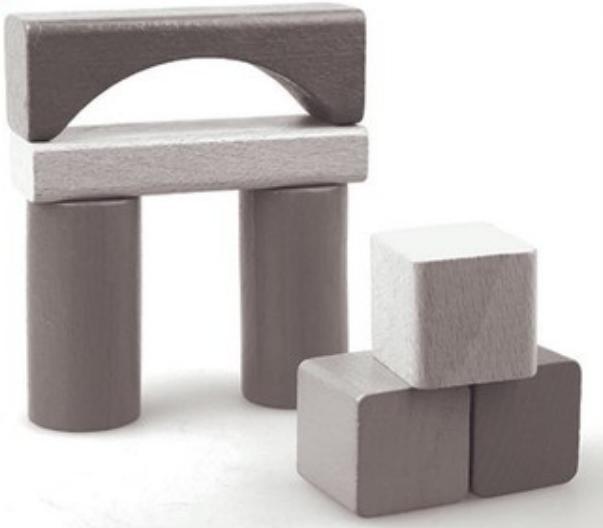
ESERCIZI COMMENTATI

Segui la risoluzione passo passo



LABORATORI CASE STUDIES

Approfondisci con i casi pratici



PREREQUISITI

- Conoscere le caratteristiche del modello E/R.
- Conoscere il concetto di vincolo.

PER COMINCIARE

1. Che cosa indica il termine entità?
2. Che cos'è la cardinalità di una relazione?
3. Che cos'è una vista?
4. Una base di dati è:
 - A un insieme di programmi
 - B un insieme di dati strutturati
 - C un insieme di routine per la gestione dei file
 - D un insieme di interfacce grafiche
5. Il modello E/R descrive:
 - A la singola vista
 - B lo schema concettuale
 - C lo schema logico
 - D lo schema fisico



CONOSCENZE

- Conoscere le caratteristiche del modello relazionale.
- Conoscere i principali tipi di operatori relazionali.
- Conoscere le operazioni dell'algebra relazionale.
- Conoscere il processo di normalizzazione e le principali forme normali.
- Conoscere i principali vincoli di integrità.



ABILITÀ

- Saper passare dal modello E/R al modello relazionale.
- Saper operare con i principali operatori relazionali.
- Saper normalizzare una relazione.
- Saper impostare dei vincoli su una relazione.



COMPETENZE

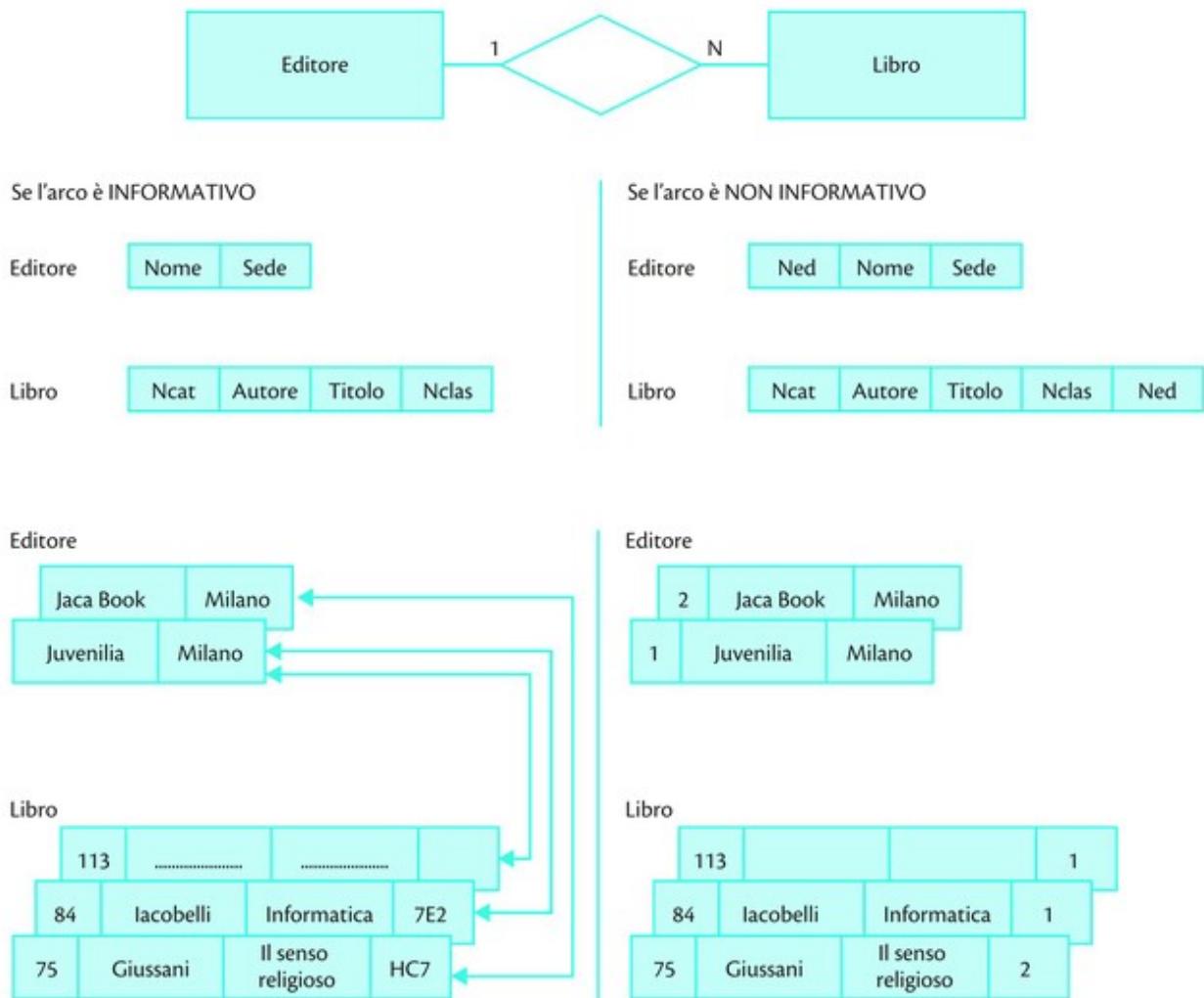
- Utilizzare le strategie del pensiero razionale negli aspetti dialettici e algoritmici per affrontare situazioni problematiche elaborando opportune soluzioni.
- Scegliere dispositivi e strumenti in base alle loro caratteristiche funzionali.
- Gestire progetti secondo le procedure e gli standard previsti dai sistemi aziendali di gestione della qualità e della sicurezza.
- Redigere relazioni tecniche e documentare le attività individuali e di gruppo relative a situazioni professionali.

1. I MODELLI LOGICI

Descriviamo ora i possibili modelli logici, cioè quei modelli che permettono una reale implementazione in uno specifico DBMS. Tutta la gestione delle basi di dati dipende da come questi dati sono rappresentati: gli stessi linguaggi dipendono in modo strettissimo dallo schema su cui devono operare.

I modelli dei dati utilizzati per descrivere lo schema possono essere classificati in base al modo in cui vengono rappresentate e trattate le associazioni, in particolare nel modo diverso in cui vengono gestiti gli archi delle associazioni nella rappresentazione grafica dello schema. L'arco si dice informativo se senza la sua presenza fisica non è possibile collegare le due entità. Si parla invece di archi non informativi se il collegamento è di tipo logico, e quindi anche senza la presenza fisica degli archi è possibile collegare le due entità (tramite l'uguaglianza di due attributi). Si può capire meglio la differenza esistente tra i due modelli analizzando l'esempio della fig. 1.

fig. 1 Esempio di arco informativo/non informativo



Nel caso di archi informativi, nel tracciato record dell'archivio Libro non è presente alcun campo che permetta di risalire ai dati dell'Editore. I collegamenti sono realizzati per mezzo dei puntatori gestiti dal DBMS indipendentemente dai campi dei record; l'assenza del collegamento fisico non ci permette quindi di conoscere l'editore di un libro.

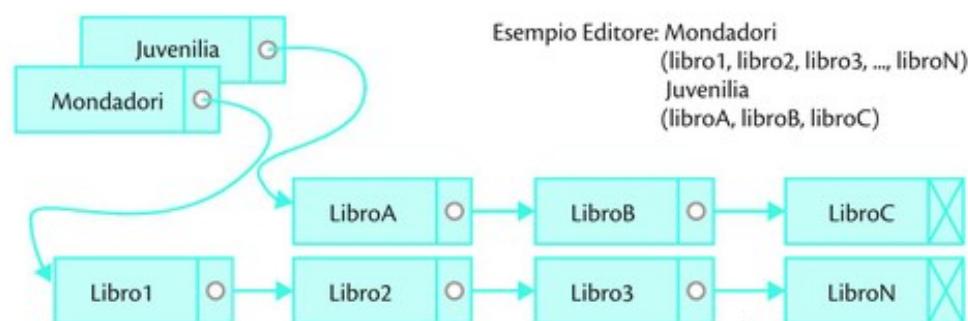
Nel caso dell'arco non informativo, il campo Ned presente nel tracciato record dell'archivio Libro determina la casa editrice da cui il libro è pubblicato. L'arco è non informativo poiché tutte le informazioni necessarie per collegare i due archivi sono già contenute nei record stessi e quindi non è necessario utilizzare puntatori fisici.

I modelli che utilizzano archi informativi sono detti **modelli a grafo**, e possono essere a loro volta suddivisi in modello gerarchico e in modello reticolare. L'uso di archi non informativi è caratteristico invece del **modello relazionale**. Un discorso a parte va invece fatto per il **modello Object-Oriented**.

Modelli a grafo

Nei modelli a grafo le relazioni sono implementate tramite puntatori (indirizzi fisici) che indicano quale istanza di entità è connessa a quella considerata. Nella relazione è individuato sempre un proprietario (padre, owner) e un posseduto (figlio, member). Se consideriamo la relazione Editore_Libro, l'entità Editore è l'entità padre, Libro è l'entità figlio. Tramite i collegamenti realizzati con i puntatori è possibile conoscere quali sono i libri pubblicati da un determinato editore. Nella [fig. 2](#) si vede che un puntatore, nell'occorrenza (istanza) Mondadori, collega questa con il suo primo libro (libro1). Ciascun libro, figlio della relazione, contiene il puntatore al fratello: libro1 contiene il puntatore a libro2, libro2 a libro3 ecc. Scorrendo sequenzialmente la catena dei fratelli si conoscono tutti i libri della casa editrice. Al momento di inserire un nuovo libro ci si deve prima posizionare sul padre Editore (inserendolo se non c'è ancora), inserire il libro desiderato e infine collegare questa istanza con i suoi fratelli o con il padre, se non ci sono fratelli.

fig. 2 Modalità di memorizzazione dei dati nel modello grafico



Modello gerarchico

Lo schema è rappresentato da una struttura ad albero, cioè da un grafo privo di cicli. L'albero, detto **albero di definizione**, ha un'organizzazione tipicamente gerarchica: ciascun nodo rappresenta una classe di entità e ciascun arco una relazione che può essere di tipo uno a molti (1:N) o uno a uno (1:1). Nella [fig. 3](#) è illustrato un esempio di schema gerarchico.

Ogni nodo ha un solo genitore, mentre un padre può avere più figli. Ogni entità figlio può essere collegata a elementi di un'altra entità figlio solo passando dall'entità superiore. Dall'esempio della [fig. 3](#) si può vedere che è possibile passare dall'entità Uffici all'entità Laboratori solo attraverso l'entità Edificio. Dato che fra le due entità è possibile definire una sola relazione, non è necessario dare i nomi alla relazione, ma è sufficiente specificare chi è il padre e chi è il figlio.

[fig. 3](#) Modello gerarchico



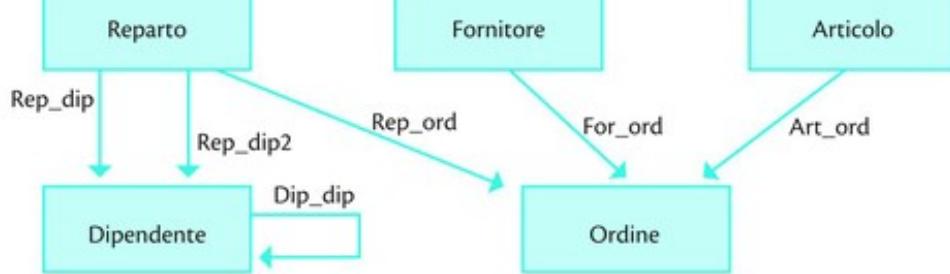
LO SAI CHE

Storicamente i modelli gerarchici sono i primi tipi di DBMS immessi sul mercato, nati per implementare in modo rapido ed efficiente situazioni aventi una reale natura gerarchica. Risultano invece lenti e macchinosi per problemi di natura diversa.

Modello reticolare

Non è richiesto che lo schema sia un albero, ma ogni tipo di entità può avere nello schema un numero qualsiasi di tipi di entità connessi a esso (lo schema è un reticolo). È possibile l'esistenza di più di una relazione tra due entità ed è per questo che le relazioni devono avere un nome. Sono permesse anche relazioni di tipo N:N e relazioni unarie (archi che partono e arrivano nello stesso nodo). Nello schema sono descritti i record-type (le entità), i data-items (gli attributi) e i set (le associazioni tra entità) in cui c'è una gerarchia tra un owner e un member. È tramite il set che si effettua la navigazione, perché un record-type può appartenere a più set. Nella [fig. 4](#) viene illustrato un esempio di schema reticolare.

[fig. 4](#) Modello reticolare



Modello relazionale

Nel modello relazionale i dati vengono rappresentati in formato tabellare. Le tabelle sono formate da colonne per gli attributi e da righe per i record ([fig. 5](#)).

Il legame esistente tra due tabelle è di tipo logico ed è realizzato tramite un attributo in comune, che nell'esempio della [fig. 5](#) è rappresentato dal campo CodEd della tabella Editori e dal campo CodEd della tabella Libri. Il modello relazionale sarà trattato più approfonditamente nelle prossime lezioni.

fig. 5 Organizzazione dei dati in forma tabellare

Libri

Ncat	autore	titolo	Nclas	CodEd
75	Manzoni	I promessi sposi	7	1
90	Leopardi	I canti	89	2
123	Pellico	Le mie prigioni	H3	1

Editori

CodEd	nome	sede
1	Mondadori	Milano
2	Juvenilia	Milano

Modello Object-Oriented

Recentemente si stanno diffondendo i sistemi basati sul modello **Object-Oriented**. Tale modello utilizza oggetti indipendenti che permettono di modellizzare in modo naturale la realtà, e non un modello della realtà. Si parla di “natural modelling” (fig. 6).

fig. 6 Natural modelling

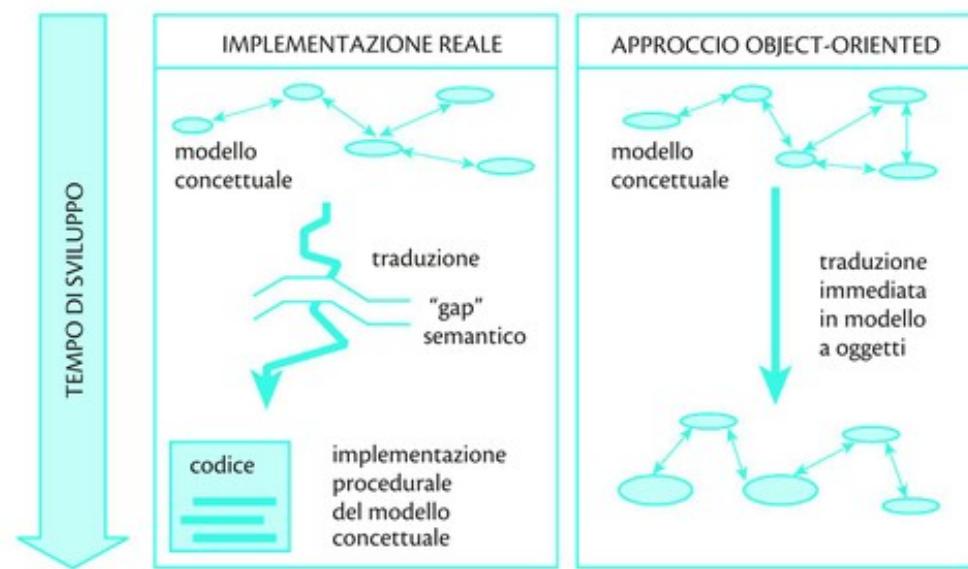


fig. 7 Oggetto



Il concetto base di questa nuova tecnologia è l'inscindibilità della struttura dei dati e delle operazioni che li manipolano: un oggetto è formato da un insieme di dati, detti proprietà, e da un insieme di operazioni, dette metodi (fig. 7). Le proprietà (o attributi) sono dati privati di un oggetto e non possono essere raggiunte esplicitamente dall'esterno se non applicando le operazioni (o metodi) associate.

Encapsulation

Il principio della **encapsulation** afferma che l'unico modo per accedere ai dati dell'oggetto è l'invocazione dei metodi: non è quindi possibile usare la struttura in modo differente da quello per cui è stata prevista. L'encapsulation può essere vista come un'estensione del concetto di modularità. Dal momento che le interazioni tra gli oggetti possono avvenire solamente tramite i metodi, risulta facilitata la costruzione incrementale di sistemi di dimensioni ragguardevoli, nonché la localizzazione di eventuali malfunzionamenti.

Ereditarietà

Un'altra caratteristica importante che differenzia i sistemi Object-Oriented da quelli tradizionali è l'ereditarietà. L'ereditarietà (*inheritance*) consente la definizione di un nuovo oggetto mediante il raffinamento: è cioè possibile creare proprietà e metodi del nuovo oggetto modificando opportunamente quelli dell'oggetto preesistente e aggiungendo ulteriori proprietà (e/o metodi) caratterizzanti. Per esempio, nell'ipotesi di possedere la definizione del tipo "persona", possiamo definire l'oggetto "impiegato" semplicemente affermando che un impiegato è una persona (ed eredita quindi tutte le proprietà e i metodi del tipo persona) e specificando solamente ciò che caratterizza gli impiegati rispetto alle persone (per esempio lo stipendio e le operazioni di assunzione, pensionamento e licenziamento). Con l'ereditarietà diventa dunque possibile riutilizzare il software prodotto senza duplicazioni e senza conoscere i dettagli di implementazione, con evidenti vantaggi in termini di produttività.

Modelli NoREL

I modelli visti fino ad ora si basano su una strutturazione rigida dei contenuti. Questo porta, prima di inserire e utilizzare i dati, a dover progettare la struttura del database, e successivamente a inserire i dati secondo le regole fornite nel database stesso. Questa è una delle caratteristiche base che hanno portato alla nascita dei database. Oggi la quantità di dati a disposizione è incrementata in modo esponenziale, e si è passati da sistemi con pochi dati mirati utilizzati da specialisti, a montagne di dati facilmente reperibili e creabili da tutti. La strutturazione dei dati diventa quindi in alcuni casi un ostacolo allo loro fruibilità, per cui sono nati database **NoREL** o **NoSQL** in cui la strutturazione è l'elemento assente, e proprio tale assenza è uno degli aspetti che maggiormente ne hanno permesso il successo. Le informazioni non trovano più posto in record e archivi, ma in oggetti completamente diversi e non necessariamente strutturati, come ad esempio documenti archiviati in collezioni.

FISSA LE CONOSCENZE

1. Come sono classificati i modelli di DBMS?
2. Come si realizza un arco informativo?
3. Come si realizza un arco non informativo?
4. Quali sono le caratteristiche del modello Object-Oriented?

2. IL MODELLO RELAZIONALE

■ Le tabelle

Il modello relazionale è stato proposto nel 1970 dal matematico inglese Edgar Codd e grazie alla sua coerenza e usabilità è diventato dagli anni Ottanta del secolo scorso quello più utilizzato per la produzione di DBMS.

La struttura fondamentale del modello relazionale è una tabella bidimensionale costituita da righe e colonne, detta **relazione**.

Le relazioni rappresentano le **entità** che si ritengono interessanti per il database: gli **attributi** delle entità rappresentano i campi delle relazioni, cioè le **colonne** delle tabelle; per ennuple (N-ple) o tuple si intendono le occorrenze delle relazioni, cioè le **righe**, o **record**, delle tabelle.

Il modello relazionale si basa sulla seguente definizione matematica di relazione:

dati due insiemi D_1 e D_2 , non necessariamente disgiunti, una relazione R definita su D_1 e D_2 è un qualunque sottoinsieme del prodotto cartesiano $D_1 \times D_2$.

ESEMPIO

Siano $D_1 = \{\text{Giovanni, Mario, Anna}\}$ e $D_2 = \{10, 25, 18\}$.

Il prodotto cartesiano $D_1 \times D_2$ è formato dalle coppie $\{(\text{Giovanni}, 10), (\text{Giovanni}, 25), (\text{Giovanni}, 18), (\text{Mario}, 10), (\text{Mario}, 25), (\text{Mario}, 18), (\text{Anna}, 10), (\text{Anna}, 25), (\text{Anna}, 18)\}$.

Una relazione è un sottoinsieme del prodotto cartesiano e pertanto, dal prodotto cartesiano appena calcolato, estraiamo solo le coppie per noi significative, quelle che rappresentano, per esempio, l'età delle persone.

Definiamo quindi la relazione Persone, formata dalle coppie $\{(\text{Giovanni}, 10), (\text{Mario}, 25), (\text{Anna}, 18)\}$, che si rappresenta in tabella come è illustrato nella [fig. 8](#).

PERSONE

Giovanni	10
Mario	25
Anna	18

fig. 8 Rappresentazione tabellare della relazione Persone

Potrebbe nascere confusione tra relazione intesa come "insieme di dati nel modello relazionale" e relazione intesa come "associazione tra entità". In questa Unità utilizzeremo il termine **tabella** invece del termine **relazione**, ricordando che sono sinonimi, così come lo sono i termini "campo" e "attributo", e i termini "record", "occorrenza" ed "ennupla".

Nella **fig. 9** sono mostrati altri esempi di tabelle relazionali.

Amici

cognome	indirizzo	città	n. telefono
Rossi	Via Roma 4	Torino	011 3572943
Bianchi	Via Torino 12	Milano	02 4372049
Verdi	Via Genova 12	Torino	011 6993524

fig. 9 Tabelle relazionali

Località

città	regione
Torino	Piemonte
Milano	Lombardia
Roma	Lazio

■ L'identificazione dei record

Per identificare i record all'interno della tabella si fa uso di **campi chiave** o di identificatori.

Un **identificatore** è un insieme di uno o più attributi che identifica univocamente una data occorrenza.

Al posto di identificatore si parla a volte di **chiave candidata**: in una tabella, possono esistere più chiavi candidate (per esempio, nella tabella Persona possono esistere due identificatori, il Codice fiscale e l'insieme di Nome, Cognome e Data_Nascita).

Si osservi che almeno una chiave candidata esiste sempre, poiché al limite può essere costituita dall'insieme di tutti gli attributi (non possono infatti esistere record uguali, cioè le righe della tabella devono essere tutte diverse).

Tra tutte le chiavi candidate viene scelta una **chiave primaria** per la memorizzazione e per effettuare i collegamenti con le altre relazioni. Normalmente ne viene scelta una tra quelle che hanno il minor numero di attributi.

Gli attributi della chiave primaria hanno la caratteristica di non potere mai assumere il valore **null** (che significa un valore non determinato), perché in tal caso non permetterebbero più di identificare una particolare tupla in una relazione.

Una **chiave esterna** è un attributo (o un insieme di attributi) che non è chiave nella tabella, ma lo è in un'altra e serve per realizzare il collegamento logico tra le tabelle.

■ Il dominio dei campi

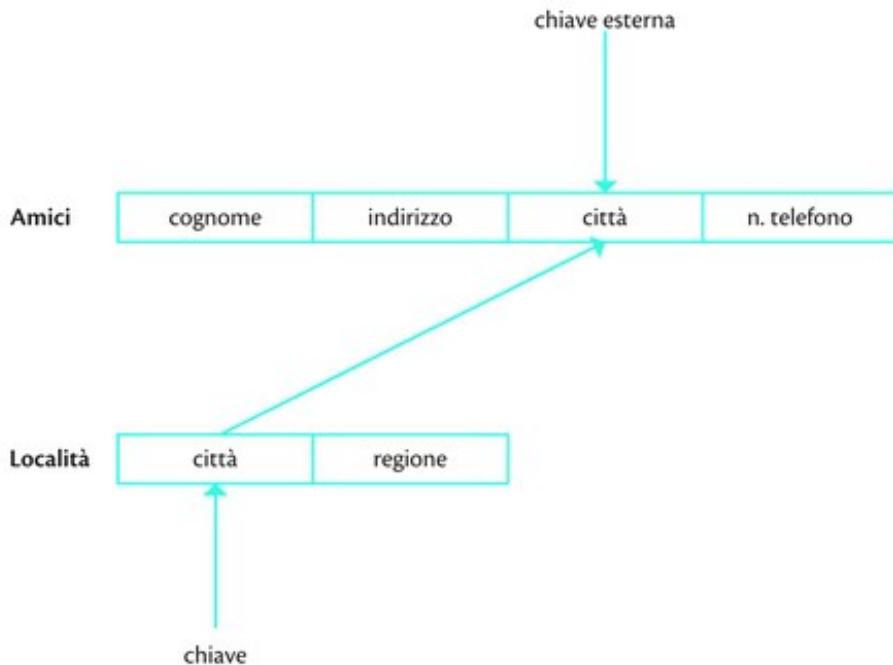
Due tabelle possono essere messe in relazione solo attraverso l'uguaglianza dei valori di un certo campo. Due valori sono confrontabili solo se appartengono allo stesso dominio.

Per **dominio** si intende l'insieme dei valori su cui è definito un certo attributo, cioè l'insieme di tutti i valori elementari che un attributo può assumere (per esempio, il campo Età potrà essere definito sul dominio degli interi che va da 0 a 150).

Il concetto di dominio è fondamentale: gli attributi di diverse relazioni definite sullo stesso dominio permettono di realizzare i collegamenti tra le diverse tabelle.

In particolare questo è possibile tramite l'uguaglianza tra chiave primaria di una tabella e chiave esterna di un'altra. Nell'esempio della [fig. 10](#) si vede come la tabella Località sia collegata alla tabella Amici, poiché in entrambe esiste l'attributo Città, il quale è chiave primaria nella tabella Località e chiave esterna nella tabella Amici.

[fig. 10](#) Chiave primaria e chiave esterna



■ Metriche

Altre caratteristiche risultano utili per definire le **metriche**, ossia le *misure quantitative*, sulle relazioni presenti in un database.

Per **grado** di una relazione si intende il numero di campi delle tabelle, mentre per **cardinalità** si intende il numero delle occorrenze presenti in un dato istante.

Nello schema della [fig. 11](#) sono riassunti e illustrati i concetti appena espressi sull'esempio della tabella Amici.

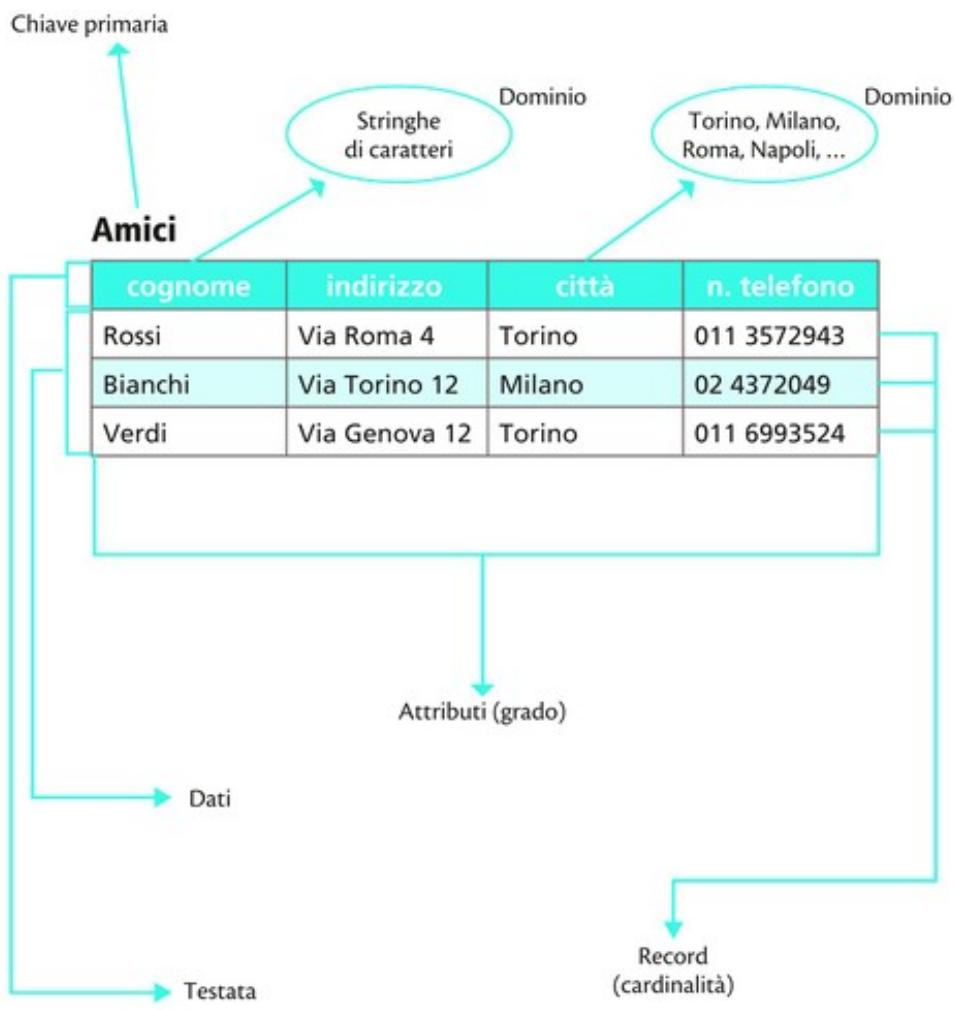


fig. 11 Caratteristiche delle tabelle

■ Altre caratteristiche

Vi è una serie di caratteristiche che permettono di definire un database relazionale. Queste sono riassunte nelle **12 regole di Codd** che riportiamo nell'approfondimento. Si parla di dodici regole, ma in realtà sono tredici, in quanto si parte dalla regola 0.

Le regole sono molto restrittive e attualmente nessun sistema le rispetta veramente tutte.

Evidenziamo, a partire dalle 12 regole di Codd, alcuni punti che riassumono le caratteristiche che devono avere le relazioni nel modello relazionale.

- Non è definito alcun ordinamento fra le N-ple, quindi due tabelle con le stesse righe, anche in ordine diverso, rappresentano la stessa relazione (fig. 12).

Auto

targa	colore	modello
AB067BX	rosso	Fiat 600
CD832AT	verde	Ford Fiesta
LD325CZ	bianco	Fiat Panda

fig. 12 Esempio di tabelle coincidenti

Automobile

targa	colore	modello
CD832AT	verde	Ford Fiesta
AB067BX	rosso	Fiat 600
LD325CZ	bianco	Fiat Panda

- Le N-pie di una relazione devono essere distinte l'una dall'altra; quindi una tabella rappresenta una relazione solo se le sue righe sono tutte diverse fra loro (fig. 13).

Studente

fig. 13 Esempio di N-uple non distinte

cognome	età	città
Rossi	18	Torino
Bianchi	17	Roma
Verdi	18	Torino
Rossi	17	Roma
Rossi	18	Torino

- È possibile che alcune informazioni non siano disponibili per tutte le N-pie della relazione: possono essere ignote o non definite. Questi attributi assumono un valore speciale, denominato valore nullo (null); tale valore non fa parte di alcun dominio e rappresenta sia un valore ignoto, sia un valore non definito.

Inoltre, si fa uso del valore null per un attributo quando questo non è significativo per gli altri attributi oppure è un valore che verrà inserito in un secondo momento.

È possibile che alcune colonne di una tabella non possano assumere valore null. Per esempio, nella relazione **Studente** (**Matricola**, **Cognome**, **Data_Nascita**, **Telefono**, **Anno_Laurea**) il telefono può essere ignoto e per uno studente non ancora laureato l'anno di laurea non è ancora definito, ma la matricola non potrà mai essere ignota (fig. 14).

Studente

fig. 14 Campi con valore ignoto o non definito

matricola	cognome	data_nascita	telefono	anno_laurea
74325	Rossi	22/06/1994	011 3786354	null
45861	Bianchi	10/08/1988	null	2013
69437	Verdi	09/03/1990	null	null

- In una relazione esiste sempre una chiave candidata, poiché al limite può essere costituita dall'insieme di tutti gli attributi.
- Tra tutte le chiavi candidate (gli identificatori) viene scelta una chiave primaria per la memorizzazione e per effettuare i collegamenti con

le altre relazioni.

Normalmente ne viene scelta una tra quelle con il minor numero di attributi.

- Gli attributi della chiave primaria non possono mai assumere il valore null (che significa un valore non determinato), perché in tal caso non permetterebbero più di identificare una particolare N-pla in una relazione.
- Una chiave esterna è un attributo (o un insieme di attributi) che non è chiave nella tabella, ma lo è in un'altra e serve per realizzare il collegamento logico tra le entità.
- Due tabelle possono essere messe in relazione solo attraverso l'ugualanza dei valori di un certo campo.
- Due valori sono confrontabili solo se appartengono allo stesso dominio.

APPROFONDIMENTO

Le 12 regole di CODD

- **Regola 0:** il sistema deve potersi definire come *relazionale, base di dati e sistema di gestione*. Affinché un sistema possa definirsi sistema relazionale per la gestione di basi di dati (RDBMS), tale sistema deve usare le proprie funzionalità *relazionali* (e solo quelle) per gestire la base di dati.
- **Regola 1:** l'informazione deve essere rappresentata sotto forma di tabelle. Le informazioni nel database devono essere rappresentate in maniera *univoca*, e precisamente attraverso valori in colonne che costituiscano, nel loro insieme, righe di tabelle.
- **Regola 2:** la regola dell'accesso *garantito* o delle *chiavi primarie*. Tutti i dati devono essere accessibili senza ambiguità. Ogni singolo valore scalare nel database dev'essere logicamente indirizzabile specificando il nome della tabella che lo contiene, il nome della colonna in cui si trova e il valore della chiave primaria della riga in cui si trova.
- **Regola 3:** trattamento sistematico del valore NULL. Il DBMS deve consentire all'utente di lasciare un campo vuoto, o con valore NULL. In particolare, deve gestire la rappresentazione di *informazioni mancanti* e quella di *informazioni inadatte* in maniera predeterminata, distinta da ogni valore consentito (per esempio, "diverso da zero o qualunque altro numero" per valori numerici), e indipendente dal tipo di dato. Queste rappresentazioni devono inoltre essere gestite dal DBMS sempre nello stesso modo.
- **Regola 4:** dizionario del modello relazionale. La descrizione della struttura del database e degli oggetti che lo compongono deve avvenire ad un *livello logico*, tramite un dizionario di metadati, e questo dizionario deve essere accessibile agli utenti del database con le stesse modalità e lo stesso linguaggio di interrogazione utilizzato per accedere ai dati.
- **Regola 5:** accessibilità dei dati. Tutti i contenuti del database devono essere accessibili attraverso almeno un linguaggio relazionale (come ad esempio l'SQL) che abbia le seguenti caratteristiche:
 - abbia una *sintassi lineare* (ovvero le cui istruzioni possono essere semanticamente interpretate con una semplice lettura da sinistra verso destra)

- possa essere utilizzato sia in forma interattiva che dall'interno di applicazioni
 - supporti operazioni di definizione e di manipolazione dei dati, le regole di sicurezza e i vincoli di integrità del database.
- **Regola 6:** aggiornamento delle viste di dati.
In un database relazionale si possono creare viste che forniscono l'accesso a specifici subset di informazioni. Laddove il contenuto in termini di dati di queste viste è concettualmente modificabile, lo deve anche essere nella pratica.
- **Regola 7:** manipolazione dei dati ad alto livello.
Da un database possiamo reperire informazioni multiple costituite da set di dati provenienti da più righe e/o più tabelle. Gli stessi set di dati, piuttosto che le singole informazioni, devono anche poter essere inseriti, aggiornati e cancellati.
- **Regola 8:** indipendenza dalla rappresentazione fisica.
La struttura logica di un database deve essere indipendente dalle strutture di memorizzazione fisica: modifiche al piano fisico (come i dati vengono memorizzati, su quali unità, con quale organizzazione, ecc.) non devono richiedere un cambiamento alle modalità di accesso al database.
- **Regola 9:** indipendenza dalla rappresentazione logica.
Le modifiche al livello logico (tabelle, colonne, righe, chiavi primarie, ...) non devono richiedere cambiamenti non giustificati alle applicazioni che utilizzano il database.
- **Regola 10:** i vincoli logici sui dati devono essere memorizzati nel database.
I vincoli di integrità propri delle entità e delle relazioni, le regole di sicurezza e le restrizioni di accesso devono essere definiti nel dizionario del database e sono quindi separati dalle applicazioni che lo utilizzano. Deve quindi essere possibile modificare tali vincoli senza inutilmente interessare le applicazioni esistenti.
- **Regola 11:** indipendenza di localizzazione.
La distribuzione di porzioni del database su una o più allocazioni fisiche o geografiche deve essere invisibile agli utenti del sistema. Le applicazioni esistenti devono continuare ad operare con successo quando i dati esistenti vengono ridistribuiti in modo diverso.
- **Regola 12:** regola di non sovversione.
Gli strumenti di accesso ai dati non devono poter annullare le restrizioni del database, per esempio aggirando i vincoli di integrità, le relazioni o le regole di sicurezza.

FISSA LE CONOSCENZE

1. Che cosa è una tabella?
2. Da dove deriva il termine modello relazionale?
3. Come si definisce una chiave primaria?
4. Che cosa indica il dominio di un attributo?
5. Che cosa indica il grado di una tabella? E la cardinalità?

3. RISTRUTTURAZIONE DELLO SCHEMA E/R

Ristrutturare lo schema

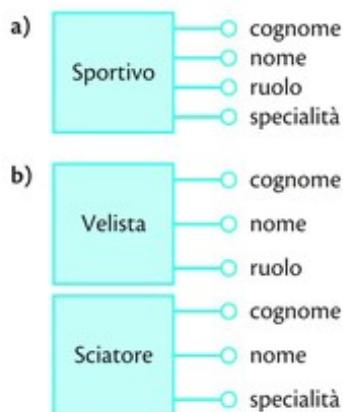
Prima di procedere alla traduzione dello schema E/R in uno schema logico relazionale è necessario effettuare una sua ristrutturazione con lo scopo di semplificarne la traduzione e ottimizzare le prestazioni. Uno schema E/R ristrutturato risulta "meno concettuale", perché tiene già conto del modello logico adottato (e quindi dei dettagli implementativi): questo ci permetterà di ottimizzare le prestazioni. Nell'attività di ristrutturazione è necessario quindi semplificare lo schema, eliminando le gerarchie e gli attributi multipli, ed effettuare un partizionamento o accorpamento di entità e relazioni.

Eliminazione delle gerarchie

Nello schema relazionale le entità e le associazioni sono direttamente rappresentabili, non è così per le gerarchie. È quindi necessario eliminarle, sostituendole con entità e associazioni. Consideriamo la gerarchia presente nella [fig. 15](#); per eliminarla senza perdere informazioni possiamo adottare metodi diversi.

1. Accorpamento delle figlie nel padre: gli attributi delle entità figlie vengono fatte migrare nell'entità padre e diventano attributi opzionali. Nell'esempio di [fig. 16a](#), ruolo è un attributo opzionale che avrà un valore nel caso di un velista, mentre risulterà null nel caso di uno sciatore. Per specialità sarà l'opposto.
2. Accorpamento del padre nelle figlie: nelle entità figlie vengono riportati gli attributi presenti nell'entità padre introducendo *ridondanza* di dati. Nel caso della [fig. 16b](#) vengono create le entità Velista e Sciatore, che avranno entrambe, oltre ai propri attributi specifici, anche quelli generali (cognome e nome).
3. Sostituzione della gerarchia con associazioni ([fig. 16c](#)). In questo caso le entità "Velista" e "Sciatore" risultano **Entità Deboli**, sono collegate ad una entità "Sportivo" (Forte) da associazioni e vengono da questa identificate esternamente.

Sono anche possibili soluzioni "ibride" in cui vengono create solo alcune nuove entità.



c)

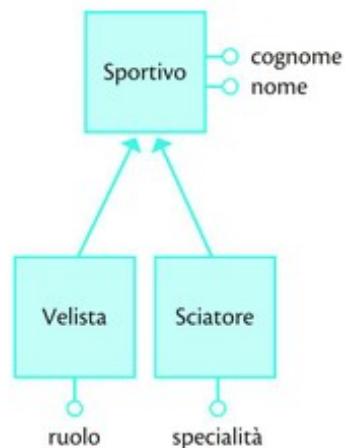
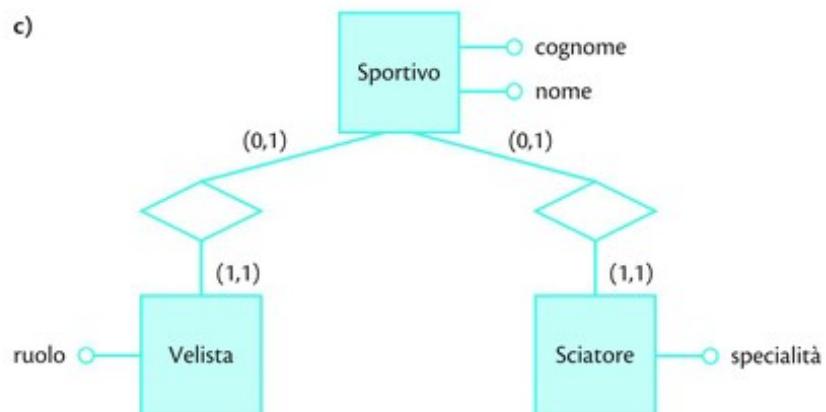


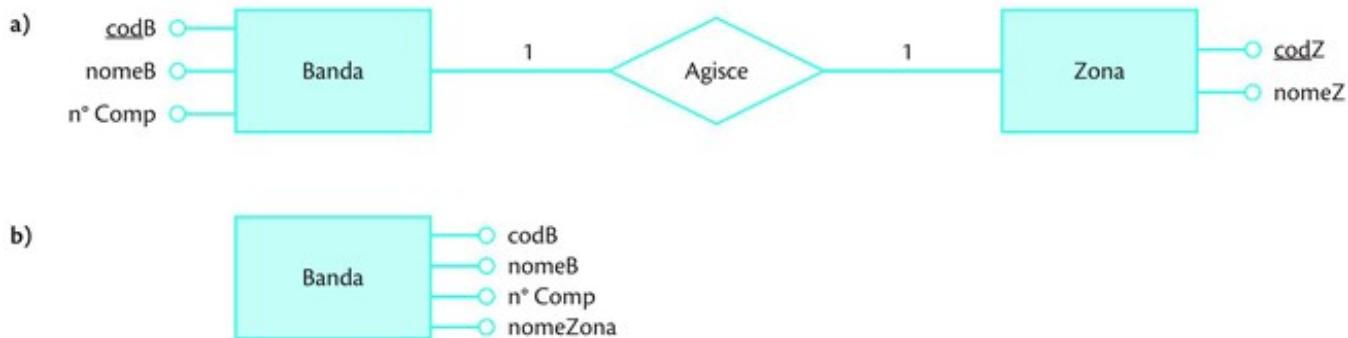
fig. 15 Gerarchia

fig. 16 Eliminazioni gerarchie

■ Partizionamento o accorpamento di entità e relazioni

In alcuni casi, prendendo in considerazione le richieste fatte alla base di dati dalle applicazioni, è conveniente accoppare delle entità che contengono informazioni che vengono richieste. Consideriamo l'esempio della [fig. 17a](#): l'entità Zona ha solo gli attributi nome e codice zona (codZ) ed è collegata con un'associazione 1:1 con Banda; possiamo quindi accorparla nell'entità Banda, aggiungendo l'attributo nomeZona ([fig. 17b](#)).

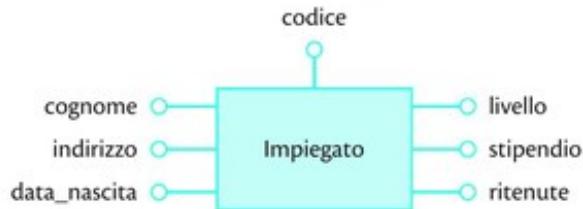
[fig. 17](#) Accorpamento di entità



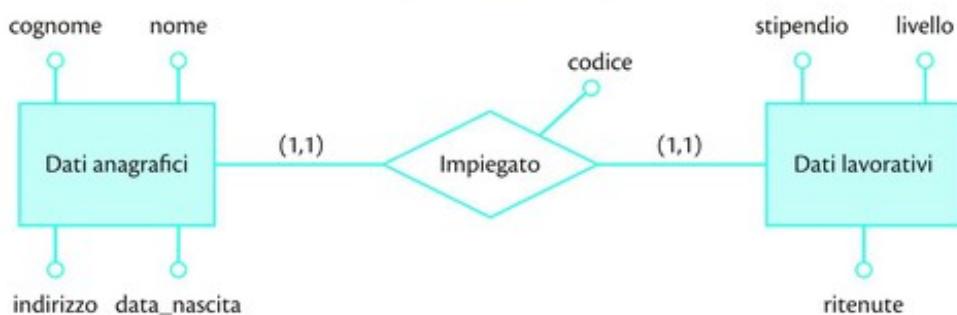
Partizionamento verticale

Supponiamo di avere la relazione impiegato, come è mostrata nella [fig. 18](#); se le nostre applicazioni accedono separatamente ai dati anagrafici o a quelli lavorativi di un impiegato, conviene separarli, perché se i record sono più piccoli in una sola lettura dal disco (operazione costosa) riesco a portare più record in memoria ([fig. 19](#)).

[fig. 18](#) Entità impiegato



[fig. 19](#) Partizionamento verticale



Partizionamento orizzontale

Supponiamo di avere lo schema rappresentato nella [fig. 20](#) e che la nostra applicazione acceda separatamente alla composizione attuale della squadra rispetto a quelle degli anni passati. Separando i due tipi di collegamento, lo schema risulta più chiaro e le operazioni più efficienti ([fig. 21](#)).

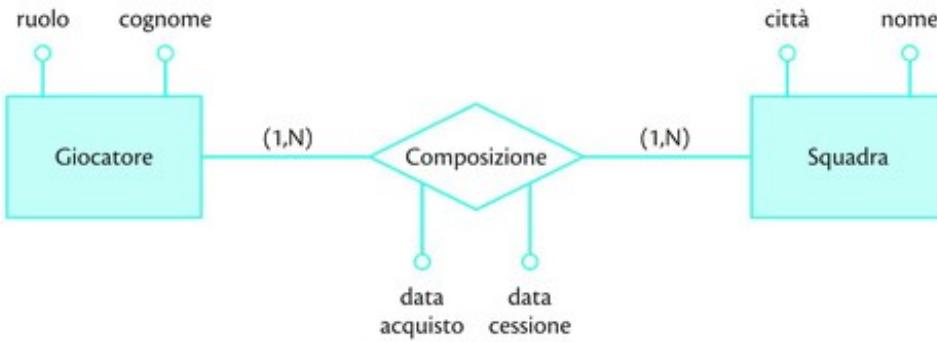


fig. 20 Schema giocatore-squadra

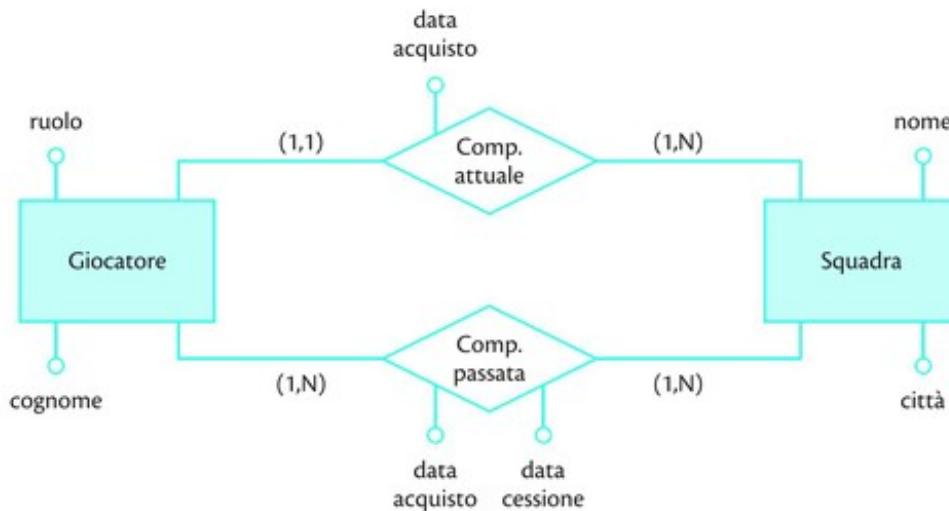


fig. 21 Partizionamento orizzontale

Eliminazione degli attributi multipli

Per eliminare gli attributi multipli che non sono rappresentabili nel modello relazionale, si crea una nuova entità con il nome dell'attributo e si inserisce una nuova relazione. Nella **fig. 22a** è mostrata l'entità libro con l'attributo multiplo autore; nella **fig. 22b** l'attributo multiplo è diventato un'entità collegata a libro da un'associazione 1:N.



fig. 22 Eliminazione attributi multipli



FISSA LE CONOSCENZE

1. Perché è necessario ristrutturare lo schema E/R prima della sua traduzione nello schema relazionale?
2. Come vengono ristrutturate le gerarchie?
3. Come si eliminano gli attributi multipli?

4. TRADUZIONE NEL MODELLO LOGICO

■ La rappresentazione delle entità

Ogni entità dello schema E/R viene rappresentata nel modello relazionale con una tabella: ogni istanza di entità rappresenta un record della tabella e ogni attributo diventa un campo del record. La chiave dell'entità diventa la chiave d'accesso per la tabella. All'entità Prodotto, rappresentata nella [fig. 23](#), corrisponde la tabella relazionale nella [fig. 24](#).

[fig. 23](#) Entità Prodotto



[fig. 24](#) Tabella relazionale

Prodotto

codice	descrizione	prezzo

Per descrivere l'entità in forma più concisa, si può scriverne il nome seguito dal nome dei suoi attributi racchiusi tra parentesi; l'attributo chiave è sottolineato. L'entità Prodotto viene in questo caso rappresentata così:

Prodotto (Codice, Descrizione, Prezzo)

La rappresentazione dettagliata dell'entità spesso avviene mediante una tabella descrittiva, nella quale sono evidenziati gli attributi e le loro caratteristiche (nome, descrizione, tipo, dimensione), come è illustrato nella [fig. 25](#).

[fig. 25](#) Descrizione della tabella Prodotto

Prodotto

nome campo	descrizione	tipo	lunghezza	chiave
Codice	codice del prodotto	stringa	3 byte	primaria
Descrizione	descrizione del prodotto	stringa	30 byte	
Prezzo	prezzo del prodotto	numerico	6 byte	

■ La rappresentazione delle associazioni

La traduzione dallo schema concettuale a quello logico delle associazioni (si dice anche *risolvere l'associazione*) risulta essere più complessa rispetto a quella delle entità. Per rappresentare le associazioni si possono effettuare scelte diverse, in base al tipo di associazione e alla destinazione d'uso.

Uno a molti

Nelle associazioni uno a molti la chiave primaria della tabella di partenza dell'associazione (quella con caratteristica 1) diventa chiave esterna (*foreign key*) dell'entità di arrivo associata, cioè l'attributo che è identificatore univoco diventa un campo nella tabella dell'entità di arrivo. Nell'esempio della [fig. 26](#), poiché una persona può possedere più auto, ma un'auto può essere posseduta da una sola persona, si farà migrare la chiave di Persona (in questo caso il codice fiscale) nella tabella Auto.

Questa diventerà chiave esterna, permettendo così il collegamento logico tra Auto e Persona.

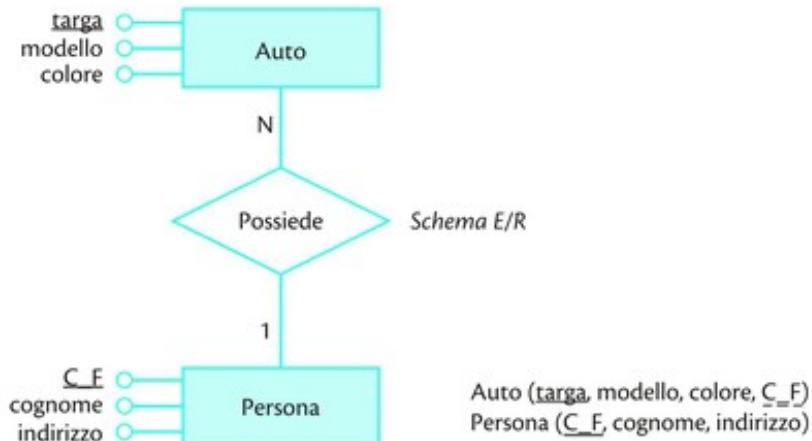


fig. 26 Soluzione delle associazioni 1:N

Dal modello concettuale vengono derivate le tabelle che rappresentano le entità e l'associazione uno a molti viene tradotta aggiungendo agli attributi dell'entità "a molti" la chiave dell'entità "a uno".

Uno a uno

Le associazioni uno a uno si risolvono facendo migrare la chiave di una delle due tabelle create nell'altra, in base alle modalità con cui si pensa verranno utilizzate.

ESEMPIO

Le entità *Persona* e *Documento identità* sono collegate tra loro (fig. 27). La loro traduzione porterà a creare due tabelle. Nella traduzione dell'associazione, la scelta potrà essere (fig. 28):

- di far migrare il numero di documento nella tabella *Persona*, se le richieste di informazioni sul numero di documento sono subordinate a quelle sulla persona (cioè prima si accede alla tabella *Persona* e poi alla tabella *Documento identità*, partendo dal suo numero di documento);
- di far migrare il codice fiscale nella tabella *Documento identità* se, viceversa, l'accesso ai dati di una persona avviene sempre dopo l'accesso ai dati del documento;
- di far migrare la chiave di ciascuna tabella nell'altra, se invece non vi sono prevalenze nell'accesso.

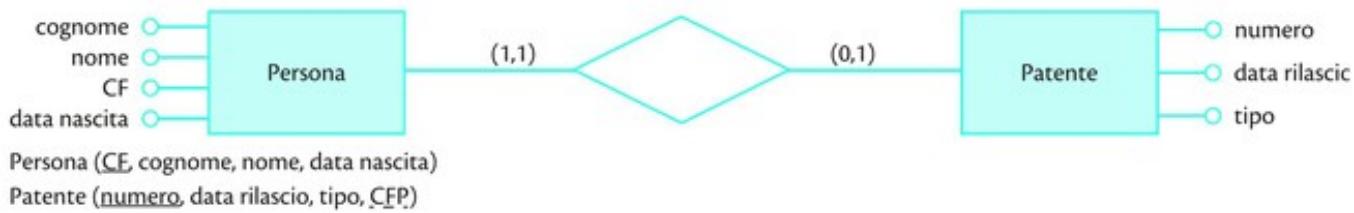


fig. 27 Relazione

- A) Documento identità (n, data rilascio, data scadenza)
Persona (CF, cognome, nome, n. documento)
- B) *Persona* (CF, cognome, nome)
Documento identità (n, data rilascio, data scadenza, CF_Persona)
- C) *Persona* (CF, cognome, nome, n. documento)
Documento identità (n, data rilascio, data scadenza, CF_Persona)

fig. 28 Possibili traduzioni del modello E/R

fig. 29 Traduzione associazione 1:1 opzionale



Molti a molti

Le associazioni molti a molti (N:N) si risolvono creando una nuova tabella (in aggiunta alle tabelle derivate dalle due entità), in cui vengono inserite le chiavi delle due entità e gli eventuali attributi dell'associazione (fig. 30).

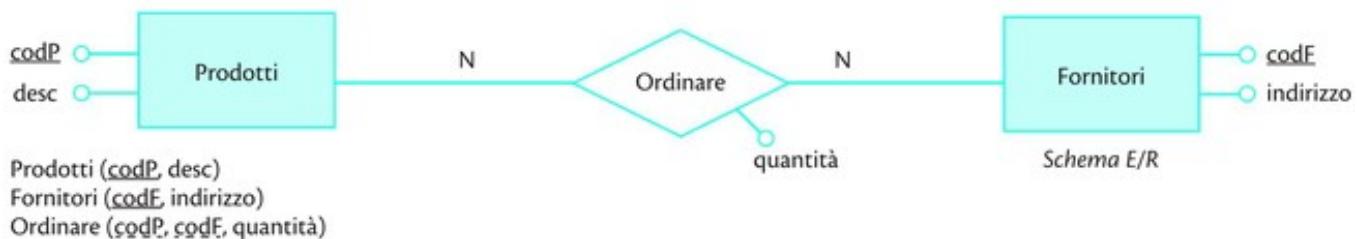


fig. 30 Schema e soluzione delle associazioni N:N

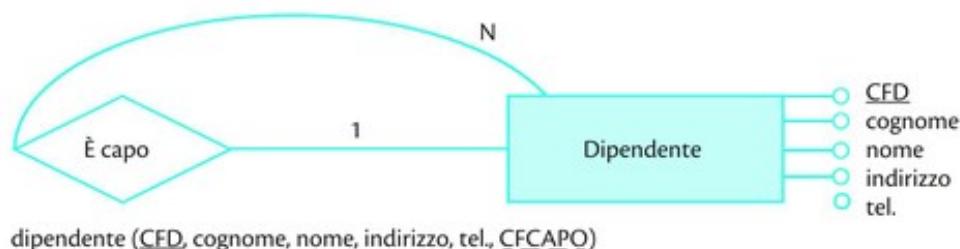
Nell'esempio riportato, poiché un fornitore può fornire più prodotti e un prodotto può essere fornito da più fornitori, si crea una nuova tabella (Ordinare) composta dalle chiavi primarie delle tabelle di partenza (CodF e CodP) e dall'attributo quantità.

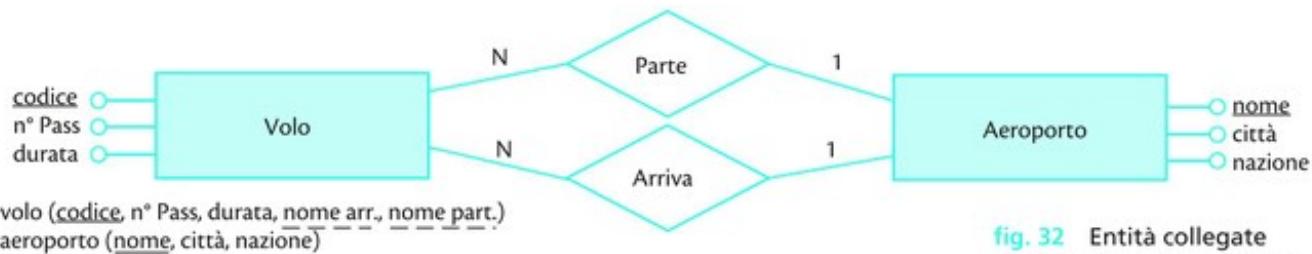
Si nota come né CodF né CodP possano essere chiave primaria della relazione Ordinare, perché non individuerebbero in modo univoco una riga della tabella. In questo caso la chiave primaria è data dalla concatenazione delle due chiavi esterne. Ogni record della tabella Ordinare indica quindi la quantità di un certo prodotto venduta da un determinato fornitore.

Dal modello concettuale vengono derivate le tabelle che rappresentano le entità e l'associazione molti a molti viene tradotta introducendo una terza tabella contenente le chiavi delle due entità e gli eventuali attributi dell'associazione.

Le regole di traduzione enunciate vengono utilizzate anche per i casi particolari, come le associazioni ricorsive ed entità collegate da associazioni diverse. Esempi di tali traduzioni sono riportati in fig. 31 e fig. 32.

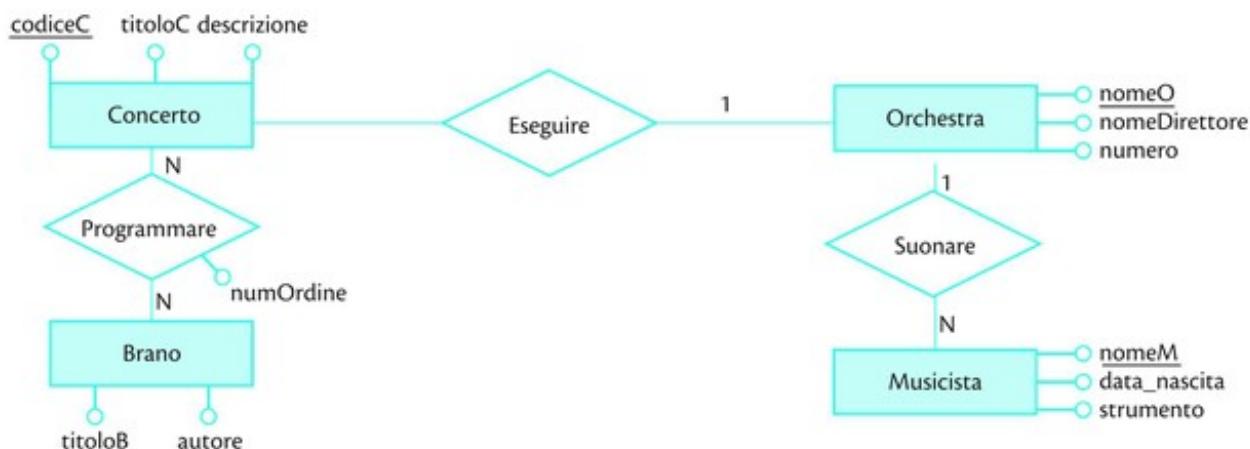
fig. 31 Traduzione di associazioni ricorsive





LABORATORIO

IL PROBLEMA Definire lo schema relazionale derivante dallo schema E/R riportato nella figura.



L'ANALISI Lo schema da tradurre non ha bisogno di essere ristrutturato; applicando le regole di derivazione, otteniamo lo schema logico relazionale.

LA RISOLUZIONE

Si possono riassumere le informazioni presenti nello schema E/R con le seguenti tabelle.
 Le entità presenti: Concerto, Orchestra, Musicista e Brano diventano altrettante tabelle contenenti gli stessi attributi delle entità.

Per quanto riguarda le relazioni, tra l'entità Orchestra e l'entità Concerto esiste la relazione uno a molti, che viene tradotta nel modello logico relazionale, inserendo la chiave primaria della tabella Orchestra (NomeO) nella tabella Concerto.

Tra l'entità Orchestra e l'entità Musicista esiste la relazione uno a molti, che viene tradotta inserendo la chiave primaria della tabella Orchestra (NomeO) nella tabella Musicista.

Tra l'entità Concerto e l'entità Brano esiste la relazione molti a molti, che viene tradotta inserendo una nuova tabella (Programmare) contenente le chiavi delle due entità a essa relazionate (CodiceC, TitoloB) e l'attributo della relazione NumOrdine.

Lo schema logico relazionale risulta pertanto:

Concerto (CodiceC, TitoloC, Descrizione, NomeO)

Orchestra (NomeO, Nome Direttore, Numero)

Musicista (NomeM, Data Nascita, Strumento, NomeO)

Brano (TitoloB, Autore)

Programmare (CodiceC, TitoloB, NumOrdine)

Più dettagliatamente avrai:

CONCERTO

nome campo	descrizione	tipo	lunghezza	chiave
CodiceC	codice del concerto	stringa	4 byte	primaria
Titolo	titolo del concerto	stringa	20 byte	
Descrizione	descrizione del concerto	stringa	50 byte	
NomeO	nome dell'orchestra	stringa	30 byte	esterna

BRANO

nome campo	descrizione	tipo	lunghezza	chiave
TitoloB	titolo del brano	stringa	50 byte	primaria
Autore	autore del brano	stringa	20 byte	

PROGRAMMARE

nome campo	descrizione	tipo	lunghezza	chiave
TitoloB	titolo del brano	stringa	50 byte	esterna
CodiceC	codice del concerto	stringa	4 byte	esterna
NumOrdine	numero d'ordine di esecuzione del brano all'interno del concerto	intero	2 byte	

ORCHESTRA

nome campo	descrizione	tipo	lunghezza	chiave
NomeO	nome dell'orchestra	stringa	30 byte	primaria
NomeDirettore	nome del direttore dell'orchestra	stringa	30 byte	
Numero	numero di elementi dell'orchestra	intero	2 byte	

MUSICISTA

nome campo	descrizione	tipo	lunghezza	chiave
NomeM	nome del musicista	stringa	30 byte	primaria
DataNascita	data di nascita del musicista	data	8 byte	
Strumento	strumento suonato dal musicista	stringa	20 byte	
NomeO	nome dell'orchestra	stringa	30 byte	esterna

RICAPITOLIAMO

PROGETTARE UN DATABASE

La progettazione delle basi dati si articola nelle seguenti fasi operative, illustrate nella **fig. 34**:

- progettazione concettuale;

- progettazione logica;
- progettazione fisica.

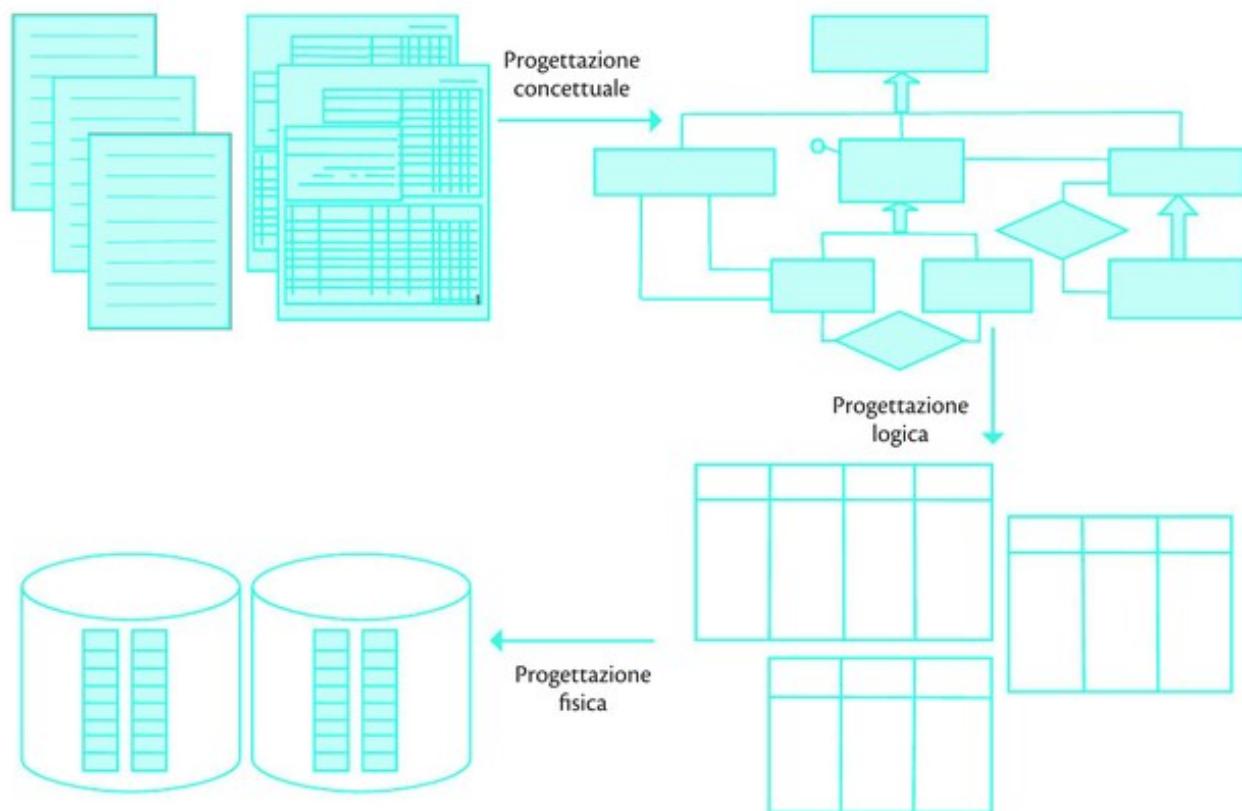


fig. 34 Fasi della progettazione

La **progettazione concettuale** ha l'obiettivo di tradurre i requisiti espressi dal cliente in una descrizione formale delle informazioni necessarie al sistema. Tale descrizione è detta **schema concettuale**, in quanto:

- è indipendente dalle caratteristiche di ogni specifico DBMS;
- la sua strutturazione dipende esclusivamente dai legami di significato che esistono tra le varie informazioni in esso contenute, non da criteri di efficienza.

La **progettazione logica** ha l'obiettivo di trasformare lo **schema concettuale** in uno schema logico conforme alle strutture proprie del DBMS scelto per la realizzazione (per esempio: schema logico gerarchico, relazionale, ecc.).

La **progettazione fisica** ha l'obiettivo di completare lo schema logico con i parametri fisici di memorizzazione e di ricerca dei dati (organizzazione dei file e degli indici), tenendo conto delle caratteristiche del DBMS e dell'ambiente hardware e software in cui la base dati sarà realizzata.

FISSA LE CONOSCENZE

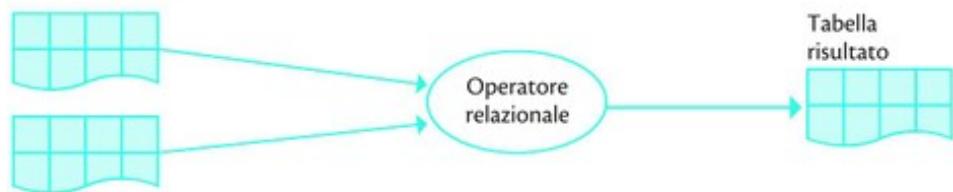
1. Come si risolve la relazione 1:1 nel modello logico?
2. Come si risolve la relazione 1:N nel modello logico?
3. Come si risolve la relazione N:N nel modello logico?

5. OPERAZIONI SULLE TABELLE RELAZIONALI

Come si è già detto, le operazioni da compiere sulle tabelle sono basate sull'**algebra relazionale**. Le operazioni di reperimento sono orientate agli insiemi di record (*set-oriented*): infatti a ogni richiesta viene fornito l'insieme delle occorrenze che soddisfano certe condizioni e non solo il primo record.

Dal punto di vista logico, il risultato di ogni operazione è una **nuova tabella** (fig. 35).

fig. 35 Operatore di algebra relazionale



Alcune di queste operazioni si basano sui concetti dell'insiemistica (unione, intersezione, differenza), mentre altre sono più specifiche. Alcune operano su una sola tabella di partenza (selezione, proiezione), mentre altre utilizzano più tabelle (prodotto cartesiano, congiunzione).

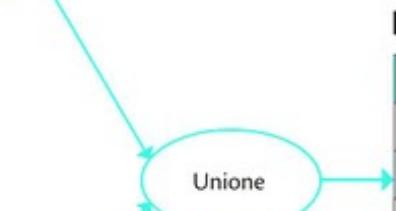
■ Operatori insiemistici

Unione: può avvenire tra due *tabelle compatibili* (cioè con lo stesso numero di campi, e questi devono avere lo stesso dominio). Il risultato dell'unione è l'insieme di tutte le occorrenze delle due tabelle esclusi i duplicati (fig. 36).

fig. 36 Esempio per l'operatore di unione

R1

Cod_F	Nome_F	città
F1	Bianchi	Torino
F2	Rossi	Milano



R

Cod_F	Nome_F	città
F1	Bianchi	Torino
F2	Rossi	Milano
F4	Gialli	Torino
F6	Blu	Genova

I duplicati vengono eliminati (F1)

R2

Cod_F	Nome_F	città
F4	Gialli	Torino
F1	Bianchi	Torino
F6	Blu	Genova

Intersezione: anche in questo caso l'operazione può essere effettuata solo su tabelle compatibili. Il risultato dell'intersezione è una tabella che contiene solo le occorrenze che sono presenti in entrambe le tabelle (fig. 37).

fig. 37 Esempio per l'operatore di intersezione

R1

Cod_F	Nome_F	città
F1	Bianchi	Torino
F2	Rossi	Milano

R2

Cod_F	Nome_F	città
F4	Gialli	Torino
F1	Bianchi	Torino
F6	Blu	Genova

Intersezione

R

Cod_F	Nome_F	città
F1	Bianchi	Torino

Differenza: anche in questo caso l'operazione può essere effettuata solo su tabelle compatibili. Il risultato della differenza è una tabella che contiene le occorrenze che sono presenti solo nella prima tabella, ma non nella seconda (fig. 38).

fig. 38 Esempi per gli operatori di differenza

Prodotto cartesiano (natural join): applicato a due tabelle restituisce una

R2

Cod_F	Nome_F	città
F1	Bianchi	Torino
F2	Rossi	Milano

Differenza

R

Cod_F	Nome_F	città
F1	Bianchi	Torino
F6	Blu	Genova

R1

Cod_F	Nome_F	città
F4	Gialli	Torino
F1	Bianchi	Torino
F6	Blu	Genova

nuova tabella le cui righe sono ottenute concatenando ogni riga della prima tabella con tutte quelle della seconda tabella (fig. 39). In genere è un insieme molto ampio e generalmente non tutte le combinazioni sono significative.

Il risultato del prodotto cartesiano è una tabella che ha $P + Q$ colonne (P indica il numero di colonne della prima tabella, Q quello della seconda) e $M * N$ righe (M indica il numero di righe della prima tabella, N quello della seconda).

Fornitori

Cod_F	Nome_F	città
F1	Bianchi	Torino
F2	Rossi	Torino
F3	Verdi	Venezia
F4	Gialli	Torino

Ordini

N_Ord	Cod_F	Cod_P	Quant
001	F1	P1	300
008	F2	P2	40

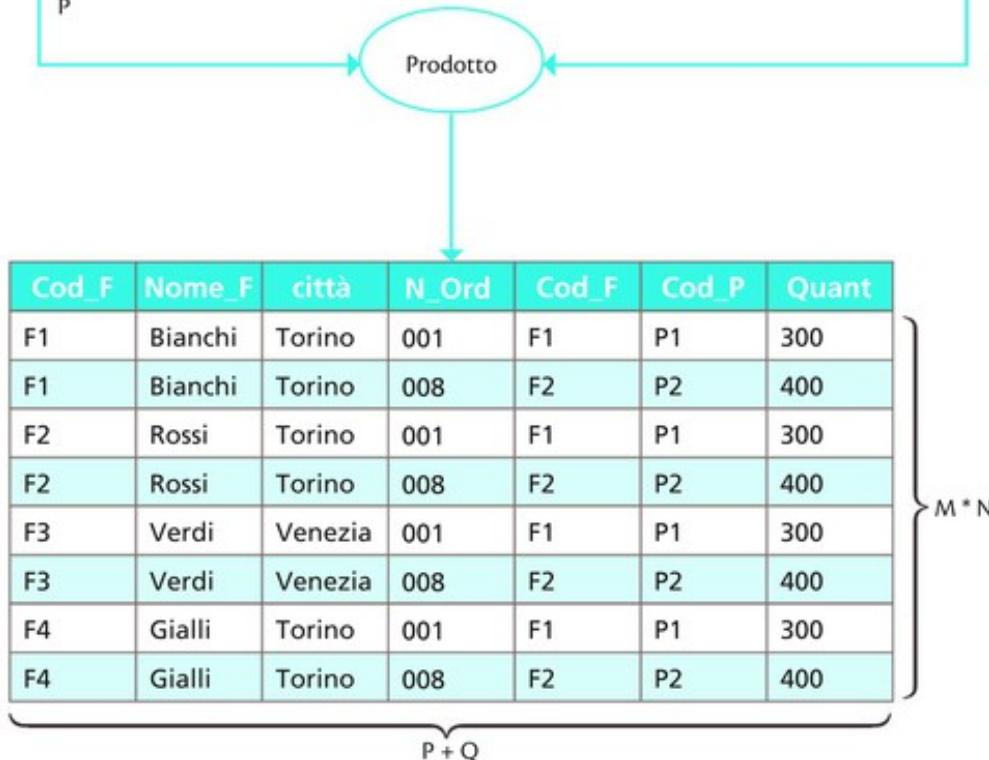
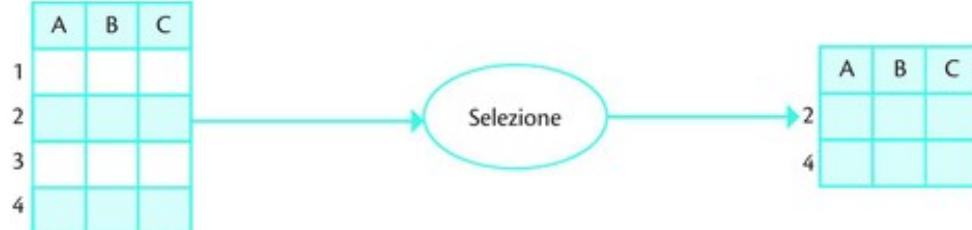


fig. 39 Esempio di prodotto cartesiano

Gli operatori relazionali agiscono su una relazione per ottenere una nuova relazione, estraendo da una tabella una sottotabella, oppure combinando tra loro due o più tabelle e generando così nuove relazioni. Nel primo caso si parla di **operatori unari** (*selezione* e *proiezione*), mentre nel secondo si ha generalmente un **operatore binario** (*congiunzione*).

Selezione (select): applicata a una tabella, fornisce come risultato l'insieme delle occorrenze che soddisfano la condizione specificata. Il risultato è una nuova tabella, che contiene tutte le colonne della tabella di partenza (stesso grado), ma ha cardinalità (numero di righe) minore o al limite uguale (fig. 40).

fig. 40 Selezione



LABORATORIO

IL PROBLEMA Si abbia il seguente schema logico relazionale:

Fornitori (Cod_F, Nome_F, città, anno_nascita).

La tabella contiene i seguenti dati.

FORNITORI

Cod_F	Nome_F	città	anno_nascita
F1	Bianchi	Torino	1971
F2	Rossi	Torino	1973
F3	Verdi	Genova	1981
F4	Gialli	Torino	1983
F5	Marrone	Napoli	1980
F6	Blu	Genova	1974

Fornire:

- i dati dei fornitori residenti a "Torino";
- i dati dei fornitori che abitano a Genova e che sono nati dopo il 1980.

L'ANALISI

Come si vede, la tabella Fornitori ha cardinalità 6 e grado 4.

- Se vogliamo ottenere i dati dei fornitori residenti a Torino, possiamo utilizzare l'operatore di selezione in questo modo:

Selezione su Fornitori dove città = "Torino"

Il risultato di questa operazione è una nuova tabella, che ha cardinalità 3 e grado 4.

Cod_F	Nome_F	città	anno_nascita
F1	Bianchi	Torino	1971
F2	Rossi	Torino	1973
F4	Gialli	Torino	1983

- Se invece vogliamo i dati dei fornitori che abitano a Genova e che sono nati dopo il 1980, possiamo impostare l'operatore logico AND nella condizione di ricerca.

Avremo pertanto:

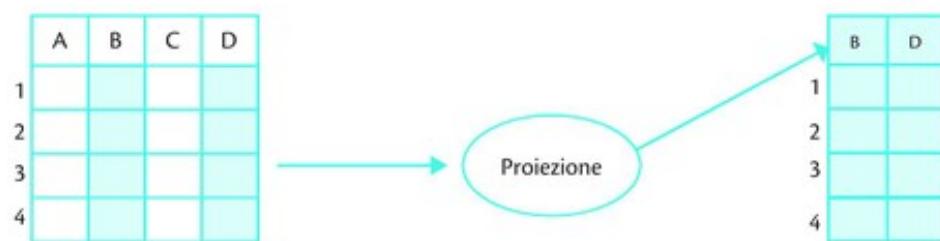
Selezione su Fornitori dove città = "Genova" and anno_nascita > 1980

che produce il risultato seguente.

Cod_F	Nome_F	città	anno_nascita
F3	Verdi	Genova	1981

Proiezione (project): applicata a una tabella, fornisce come risultato una nuova tabella, che contiene tutte le righe della tabella di partenza (stessa cardinalità), ma con le sole colonne indicate (grado inferiore) ([fig. 41](#)).

fig. 41 Proiezione



LABORATORIO

IL PROBLEMA Dallo schema logico relazionale e la tabella precedenti, ottenere:

- il codice di tutti i fornitori;
- il codice e il nome dei fornitori.

L'ANALISI

- Se vogliamo ottenere il codice di tutti i fornitori contenuti nella tabella Fornitori dell'esempio precedente, possiamo impostare l'operatore di proiezione nel seguente modo:
Proiezione su Fornitori di Cod_F

Il risultato di questa operazione è una nuova tabella, che ha cardinalità 6 e grado 1.

Cod_F
F1
F2
F3
F4
F5
F6

- Volendo invece ottenere il codice e il nome dei fornitori, possiamo impostare l'interrogazione per mezzo della seguente proiezione:

Proiezione su Fornitori di Cod_F, Nome_F

La nuova relazione ottenuta è espressa dalla nuova tabella.

Cod_F	Nome_F
F1	Bianchi
F2	Rossi
F3	Verdi
F4	Gialli
F5	Marrone
F6	Blu

Congiunzione (join): si effettua tra due tabelle che hanno un attributo in comune (cioè definito sullo stesso dominio). È un sottoinsieme del prodotto cartesiano a cui è stata applicata una clausola di selezione sull'uguaglianza dell'attributo comune. Il risultato di una join è una tabella che ha grado $P + Q - 1$: P indica il grado della prima tabella, Q quello della seconda, mentre l'attributo comune usato per la join compare una sola volta. La congiunzione qui descritta è detta *equi join*, poiché è stata formulata una condizione di uguaglianza sull'elemento comune. Con la equi join (che viene anche chiamata *inner join*) nella tabella risultante vengono inserite le righe delle due tabelle che trovano corrispondenza.

LABORATORIO

IL PROBLEMA Consideriamo le tabelle Proprietari e Auto collegate logicamente dall'attributo Cod_P (codice del proprietario).

PROPRIETARI

Cod_P	cognome
P45	Rossi
M98	Bianchi
D33	Verdi

AUTO

targa	Cod_P
AY888BD	P45
BH754AJ	D33
DK712DW	M98
BZ876AA	D33

Ottenerne l'elenco dei proprietari e delle targhe delle auto da loro possedute.

L'ANALISI Per ottenere l'elenco dei proprietari e le targhe delle auto da loro possedute, possiamo impostare l'operazione join secondo la seguente sintassi:

Giunzione su Proprietari, Auto dove Proprietari.Cod_P = Auto.Cod_P

Il risultato di questa operazione è una nuova tabella, che ha cardinalità 3 e grado 4.

Cod_F	cognome	targa
P45	Rossi	AY888BD
M98	Bianchi	DK712DW
D33	Verdi	BH754AJ
D33	Verdi	BZ876AA

Oltre alla equi join è possibile avere altri tipi di join.

- *Outer join:* è un'operazione di congiunzione tra tabelle (fig. 42), in cui nella tabella risultante compaiono non solo le righe che trovano corrispondenza, ma tutte le righe delle tabelle di partenza (fig. 42a). Le tuple che non hanno controparte vengono completate con valori Null.
- *Left join:* nella tabella risultante compaiono tutte le righe della prima tabella e quelle della seconda a esse correlate (fig. 42b).

- **Right join:** nella tabella risultante compaiono tutte le righe della seconda tabella e quelle della prima a esse correlate (fig. 42c).

Parti

Cod_F	Sede_Dispo
P1	Torino
P7	Genova

Fornitori

Cod_F	città
F1	Torino
F2	Milano
F4	Torino

Cod_F	città	Cod_P
F1	Torino	P1
F2	Milano	
F4	Torino	
	Genova	P7

a. Outer join

Cod_F	città	Cod_P
F1	Torino	P1
F2	Milano	
F4	Torino	

b. Left join

Cod_F	città	Cod_P
F1	Torino	P1
F2	Milano	
F4	Torino	

c. Right join

fig. 42 Operatore join ed esempi di outer, left e right join.

Se dalle relazioni della **fig. 43** sorgesse la necessità di conoscere quali studenti non hanno ancora superato esami, la equi join tra Studenti ed Esami_Sostenuti non sarebbe sufficiente, perché le matricole degli studenti che non hanno superato esami non compaiono fra le righe della tabella Esami_Sostenuti e quindi neppure nel risultato della join.

Studenti

matricola	cognome	nome
A1	Rossi	Mario
A2	Verdi	Michela
A3	Bianchi	Giovanni
A4	Neri	Giuseppe

Esami

Cod_Esame	materia
ES001	analisi
ES002	fisica
ES003	informatica
ES004	geometria

Studenti

matricola	cognome	voto
ES001	A1	25
ES001	A2	30
ES002	A1	30
ES002	A4	18
ES003	A1	30

fig. 43 Risultato della left join

Utilizzando la left join fra le tabelle Studenti ed Esami_Sostenuti nella tabella risultante si hanno, oltre alle righe che trovano corrispondenza, anche i record di Studenti che non trovano corrispondenza con Esami_Sostenuti. Nella **fig. 44** è riportato il risultato di questa operazione.

fig. 44 Esempi di schema

matricola	cognome	nome	Cod_Esame	voto
A1	Rossi	Mario	ES001	25
A1	Rossi	Mario	ES002	30
A1	Rossi	Mario	ES003	30
A2	Verdi	Michela	ES001	30
A4	Neri	Giuseppe	ES002	18
A3	Bianchi	Giovanni		

Un altro particolare tipo di join è la self join, che effettua la join di una tabella su se stessa.

Esaminando la tabella della [fig. 45](#) si vogliono conoscere le coppie delle persone sposate.

Cod_Fisc	cognome	nome	Cf_Coniuge
CF1	Rossi	Mario	CF3
CF2	Bruni	Riccardo	CF5
CF3	Riva	Maria	CF1
CF5	Benna	Luisa	CF2
CF7	Verdi	Giovanni	

[fig. 45](#) Tabella di esempio

Per risolvere questo problema bisogna congiungere la tabella Persona con se stessa, perché il codice fiscale del coniuge (Cf_Coniuge) è anche il codice fiscale (Cod_Fisc) di una persona.

Per non creare ambiguità, l'operazione join è effettuata sulla tabella Persona e sulla tabella T1, che è semplicemente un altro modo di chiamare la tabella Persona (è un alias).

Nella [fig. 46](#) è rappresentata la tabella risultante della self join.

Cod_Fisc	cognome	nome	Cf_Coniuge	cognome	nome	Cf_Coniuge
CF1	Rossi	Mario	CF3	Riva	Maria	CF1
CF2	Bruni	Riccardo	CF5	Benna	Luisa	CF2
CF3	Riva	Maria	CF1	Rossi	Mario	CF3
CF5	Benna	Luisa	CF2	Bruni	Riccardo	CF5

[fig. 46](#) Tabella di self join

Le operazioni descritte, a seconda dei linguaggi e dei sistemi, vengono tradotte in modo differente. In alcuni casi è necessario un comando per ogni operazione, mentre in altri è possibile effettuare più operazioni con un'unica istruzione o combinando comandi elementari.

Oltre a queste operazioni in genere vengono fornite anche altre funzionalità, come operazioni di somma, media e così via.

FISSA LE CONOSCENZE

- Quali sono i principali operatori insiemistici?
- Che cosa restituisce l'operatore di selezione?
- Che cosa restituisce l'operatore di proiezione?
- Che cosa restituisce l'operatore join?
- Qual è la differenza tra left join e right join?

6. ALGEBRA RELAZIONALE

L'algebra relazionale è un linguaggio di interrogazione. Ciascun operatore applicato a una o più tabelle produce come risultato un'altra tabella relazionale.

Nell'algebra relazionale, oltre agli operatori insiemistici, vengono implementati gli operatori relazionali nel seguente modo.

Selezione

L'operazione di selezione si rappresenta con la lettera greca σ (sigma), indicando a pedice la condizione di selezione. L'espressione $\sigma_p T$ indica la selezione delle righe della tabella T che rispecchiano il predicato P.

Esempio: $\sigma_{\text{eta} < 18} \text{Studente}$

indica la selezione di tutti gli studenti minorenni.

Proiezione

L'operazione di proiezione si rappresenta con la lettera greca π (pi), indicando a pedice la lista degli attributi selezionati. L'espressione $\pi_L T$ indica la proiezione delle colonne della tabella T specificate nella lista L.

Esempio: $\pi_{\text{nome}, \text{cognome}} \text{Studente}$

indica la proiezione dei campi nome e cognome della tabella Studente.

Congiunzione

L'operazione di congiunzione si rappresenta con il simbolo \bowtie . L'espressione $T1_{PK} \bowtie T2_{FK}$ indica la congiunzione della tabella T1 con la tabella T2 sugli attributi PK di T1 e FK di T2.

Esempio: $\text{Proprietario}_{CF} \bowtie \text{Auto}_{CFP}$

indica la congiunzione della tabella proprietario su CF e la tabella Auto su CFP.

LABORATORIO

IL PROBLEMA Considera il seguente database relazionale di una banca utilizzato per la gestione dei prestiti e dei mutui dei clienti:

cliente (cf, nome, cognome, numconto)
mutuo (idmutuo, importo, data, n.rate, cfc)

Utilizzando l'algebra relazionale, risovi le seguenti query.

1. Idmutuo e importo dei mutui contratti dal cliente con codice fiscale "BRCLST66A19F432V".

- Elenco dei clienti (codice fiscale) che hanno contratto un mutuo con importo superiore a 100.000 euro.
- Nome e cognome dei clienti che hanno contratto un mutuo in data 25/05/2019.
- Numero di rate e importo dei mutui stipulati in data 11/09/2019.
- Elenco dei conti dei clienti che hanno stipulato un mutuo nel 2018.
- Elenco dei clienti che hanno stipulato mutui con un numero di rate superiore a 10.

L'ANALISI

- Si tratta di una proiezione di idmutuo e importo sulla tabella risultante da una selezione dei mutui contratti dal dato cliente:

$\pi_{\text{idmutuo}, \text{importo}} (\sigma_{\text{cfc}=\text{"BRCLST66A19F432V"}} \text{ mutuo})$

- Dobbiamo fare una selezione dei mutui con importo superiore a 100.000 euro e visualizzare attraverso una proiezione il loro codice fiscale:

$\pi_{\text{cfc}} (\sigma_{\text{importo}>100.000})$

- In questa richiesta è necessario fare una congiunzione tra le due tabelle, perché ci vengono chiesti il nome e cognome del cliente:

$\pi_{\text{nome}, \text{cognome}} (\sigma_{\text{data}=\text{"25/05/2019"}} (\text{cliente}_{\text{cf}} \bowtie \text{mutuo}_{\text{cfc}}))$

- Dobbiamo fare una proiezione su n.rate e importo sulla tabella risultante da una selezione di tutti i mutui stipulati nella data richiesta:

$\pi_{\text{n.rate}, \text{importo}} (\sigma_{\text{data}=\text{"11/09/2019"}} \text{ mutuo})$

- Per esaudire questa richiesta è necessario fare una congiunzione tra le due tabelle, dal risultato estrarre i valori la cui data è compresa tra l'1 gennaio e il 31 dicembre del 2018 e poi visualizzare solo il numero di conto.

$\pi_{\text{numconto}} (\sigma_{\text{data}>=\text{"01/01/2018"} \text{ and } \text{data} <=\text{"31/12/2018"}} (\text{cliente}_{\text{cf}} \bowtie \text{mutuo}_{\text{cfc}}))$

- È necessario fare una congiunzione tra le due tabelle, dal risultato estrarre i mutui con un numero di rate superiori a 10 e poi visualizzare solo il nome e cognome dei clienti.

$\pi_{\text{nome}, \text{cognome}} (\sigma_{\text{n.rate}>10} (\text{cliente}_{\text{cf}} \bowtie \text{mutuo}_{\text{cfc}}))$

FISSA LE CONOSCENZE

- Che cosa è l'algebra relazionale?
- Quali sono gli operatori insiemistici?
- Quando può avvenire l'unione di due tabelle?
- Quali righe contiene la differenza di due tabelle?
- Che cosa contiene il prodotto cartesiano di due tabelle?

7. NORMALIZZAZIONE

■ Il concetto di normalizzazione

Lo schema, definito dalla progettazione concettuale, deve essere tradotto e ottimizzato, cioè depurato da gran parte delle *anomalie* di gestione che si possono verificare.

Si distinguono tre tipi di anomalia:

- *anomalia di inserimento*: se nell'inserire un nuovo record in una tabella si è costretti a inserire informazioni già presenti nel DB;
- *anomalia di cancellazione*: se nel cancellare un record si è costretti a cancellare informazioni che possono essere ancora utili nel DB;
- *anomalia di aggiornamento*: se dovendo aggiornare un record si è costretti ad aggiornarne molti altri.

Per chiarire meglio questi concetti, consideriamo le anomalie che possono verificarsi aggiornando la seguente tabella:

codProdotto	prezzo	fornitore	indirizzo	città	quantità fornita
Quaderni	1,30	Rossi	via Roma 10	Genova	100
Pennarelli	3,10	Rossi	via Roma 10	Genova	50
Matite	1,10	Verdi	via Genova 8	Torino	10.000
Penne a sfera	1,20	Verdi	via Genova 8	Torino	10.000
Colori a olio	5,50	Bruni	via Milano 10	Pisa	10

- Anomalia di inserimento: per inserire un nuovo ordine bisogna inserire nuovamente i dati anagrafici del fornitore (indirizzo, città) e del prodotto (prezzo).
- Anomalia di cancellazione: cancellando il record relativo all'ordine del prodotto Colori a olio si cancellano anche le informazioni relative al fornitore Bruni.
- Anomalia di aggiornamento: aggiornando l'indirizzo del fornitore Rossi, bisogna aggiornare due tuple.

La **normalizzazione** è l'insieme di criteri di progettazione di un database relazionale diretto a prevenire l'insorgere di tali anomalie.

La **normalizzazione** è il procedimento che trasforma successivamente le relazioni di partenza, suddividendole in altre più piccole aventi lo stesso contenuto di informazione.

Sono stati definiti diversi gradi di normalizzazione (secondo alcuni testi sono 4, secondo altri 5) a cui si fanno corrispondere le forme normali. In una trattazione rigorosa, una tabella non può essere considerata relazionale se non è normalizzata, cioè almeno in prima forma normale.

Nel corso del tempo però ci si è resi conto che la rigorosità e la purezza

matematica di una relazione non corrispondono necessariamente a una base di dati efficiente e si sono quindi sviluppati altri metodi di progettazione.

Di seguito, completiamo la descrizione del modello relazionale, riportando le definizioni delle prime tre forme normali.

■ La prima forma normale

Una relazione si dice **normalizzata** o in **prima forma normale (1FN)**, se tutti i suoi attributi hanno un dominio semplice. Non sono ammessi gruppi e ripetizioni.

La 1FN implica che ogni informazione deve essere "atomica", cioè un campo deve contenere una e una sola informazione. I campi devono contenere sempre un tipo di dato "semplice" (stringa, numero, binario ecc.) e mai aggregazioni (insieme, vettore).

La tabella Dipendenti nella [fig. 47](#) non è normalizzata, perché il campo Figli non è elementare e, inoltre, vi sono dei gruppi ripetuti (i figli di Rossi e di Bianchi). Affinché la tabella sia nella prima forma normale, è necessario trasformarla nella tabella Figli dipendenti.

Dipendenti

nome dip.	figli	
	nome	età
Rossi	Ugo	10
	Andrea	7
Bianchi	Gianni	6
	Maria	3
	Luca	1
Verdi	Pia	8

Figli dipendenti

dipen.	figlio	età
Rossi	Ugo	10
Rossi	Andrea	7
Bianchi	Gianni	6
Bianchi	Maria	3
Bianchi	Luca	1
Verdi	Pia	8

fig. 47 Tabella di figli dipendenti

■ Dipendenze funzionali

Per formalizzare i problemi visti, si introduce un nuovo tipo di vincolo, la **dipendenza funzionale**.

Consideriamo una relazione R nella quale siano definiti almeno due attributi X e Y. In R vale la **dipendenza funzionale di Y da X** (e si indica con $X \rightarrow Y$) se per ogni coppia di tuple t1 e t2 di R con gli stessi valori su X, t1 e t2 hanno gli stessi valori anche su Y. Si dice anche che **X determina funzionalmente Y** o ancora che **X è un determinante per Y**.

Chiariamo meglio questo concetto con un esempio: consideriamo la relazione Studente e stabiliamo che nella relazione la chiave primaria sia Matricola: Studente (Matricola, Nome, Telefono, Corso, Voto).

In essa si possono evidenziare le seguenti dipendenze funzionali:

- Matricola → Nome
- Matricola → Telefono
- Matricola, Corso → Voto

Nome ha una dipendenza funzionale da Matricola, perché a ogni Nome corrisponde una Matricola. Telefono dipende funzionalmente da Matricola, perché a ogni Telefono corrisponde una Matricola. Voto dipende funzionalmente da Matricola, Corso, perché a ogni Voto corrisponde l'insieme di attributi Matricola, Corso.

■ La seconda forma normale

La seconda forma normale si applica alle tabelle che hanno la chiave primaria composta da più attributi: è richiesto che tutti gli attributi di una riga dipendano dall'intera chiave primaria e non solo da una parte di essa.

Una relazione si dice in **seconda forma normale (2FN)** se è in 1FN e tutti i suoi attributi che non appartengono alla chiave dipendono funzionalmente e completamente dall'intera chiave; non possono esistere attributi che dipendono solamente da una parte della chiave.

Per esempio nella relazione Campionati, che contiene i dati dei giocatori che hanno segnato reti nei campionati, l'attributo Reti, che indica il numero di reti segnate da un giocatore in un campionato, dipende dalla chiave composta Nome-Anno, mentre l'attributo Luogo, che indica il luogo di nascita del giocatore, è dipendente solo dall'attributo Nome, che è un sottoinsieme della chiave.

Campionati

nome	luogo	anno	reti
Rossi Mario	Firenze	1998/1999	3
Rossi Mario	Firenze	1999/2000	4
Conti Bruno	Roma	1998/1999	6

La relazione non è in 2FN e presenta le seguenti anomalie:

- *anomalia di inserimento*: non è possibile inserire un nuovo giocatore sino a quando non ha segnato reti in un campionato;
- *anomalia di cancellazione*: cancellando la terza riga della tabella, si perdono le informazioni del giocatore Conti Bruno;
- *anomalia di aggiornamento*: aggiornando il luogo di nascita del giocatore Rossi, bisogna aggiornare due tuple.

Affinché la relazione Campionati sia in 2FN, bisogna eliminare dalla tabella Campionati l'attributo Luogo e aggiungere la nuova tabella Giocatori, che contiene i campi Nome e Luogo.

Giocatori

nome	luogo
Rossi Mario	Firenze
Conti Bruno	Roma

Campionati

nome	reti	anno
Rossi Mario	3	1998/1999
Rossi Mario	4	1999/2000
Conti Bruno	6	1998/1999

■ La terza forma normale

In una tabella in terza forma normale tutte le dipendenze tra colonne devono essere basate sulla chiave primaria; vengono eliminate le dipendenze funzionali transitive di un attributo non chiave da un altro attributo anch'esso non chiave.

Una relazione si dice in **terza forma normale (3FN)** se è in 2FN e tutti i suoi attributi che non appartengono alla chiave dipendono direttamente dalla chiave; non possono esistere attributi non chiave che dipendono funzionalmente da altri attributi non chiave.

Esaminiamo la relazione Fattura.

Fattura

numero	cliente	ragione_sociale	importo
001	Rossi	001-344	700
002	Rossi	001-344	600
003	Verdi	022-455	550
004	Rossi	001-344	1800

In questa relazione, che ha come chiave il campo Numero, è presente il campo Ragione_Sociale, che non dipende dalla chiave primaria, bensì da un altro campo, Cliente, che non è chiave: la tabella non è in terza forma normale, data la presenza della dipendenza transitiva:

Ragione_Sociale → Cliente → Numero

La relazione Fattura presenta le seguenti anomalie:

- **anomalia di inserimento:** non è possibile inserire la ragione sociale relativa a un cliente sino a che quest'ultimo non abbia emesso una fattura;
- **anomalia di cancellazione:** cancellando la terza riga della tabella, si perde la ragione sociale del cliente Verdi;
- **anomalia di aggiornamento:** se varia la ragione sociale di Rossi, occorre aggiornare tre tuple.

Anche la terza forma normale può essere ottenuta con un procedimento di scomposizione, separando dalla relazione di partenza il sottoinsieme

di attributi dipendente transitivamente dalla chiave primaria. Per rendere la relazione Fattura in terza forma normale, bisogna eliminare dalla tabella Fattura l'attributo Ragione_Sociale e aggiungere la nuova tabella Cliente, che contiene i campi Cliente e Ragione_Sociale.

Fattura

numero	cliente	importo
001	Rossi	700
002	Rossi	600
003	Verdi	550
004	Rossi	1800

Cliente

cliente	ragione_sociale
Rossi	001-344
Verdi	022-455

La terza forma normale può essere espressa secondo la formulazione di Boyce-Codd.

Si dice che una relazione è in BCNF (Forma normale di Boyce-Codd o Boyce-Normal Form) se è in prima forma normale (1FN) e ogni determinante è una chiave candidata, cioè ogni attributo dal quale dipendono altri attributi può essere una chiave.

LABORATORIO

IL PROBLEMA Dato il seguente database relazionale, modificarlo in modo che le tabelle siano tutte in 3FN.

Libri (codiceISBN, titolo, autore, telefono_autore, prezzo, codice-editore)
Editore (codice, nome, indirizzo, telefono)

L'ANALISI La tabella Libri non è in 3FN, perché l'attributo telefono_autore dipende da autore, che a sua volta dipende da codiceISBN (chiave primaria). Per renderla in 3FN, bisogna eliminare la dipendenza transitiva, costruendo una nuova tabella Autore.

La tabella Editore è invece in 3FN. Lo schema del database corretto sarà il seguente:

Libri (codice ISBN, titolo, cod-autore, prezzo, codice-editore)
Autore (codice, telefono)
Editore (codice, nome, indirizzo, telefono)

LABORATORIO

IL PROBLEMA Dato il seguente database relazionale, modificarlo in modo che le tabelle siano tutte in 3FN.

Ordine (codiceO, quantità, prezzo, fornitore, telefono_fornitore)

Magazzino (NumeroM, azienda, località, capacità_massima, indirizzo_azienda, telefono_azienda)

L'ANALISI La tabella Ordine è in 2FN, ma non è in 3FN, perché c'è una dipendenza transitiva (telefono_fornitore dipende da fornitore e non da codiceO), per cui deve essere divisa in due tabelle diverse:

Ordine (codiceO, quantità, prezzo, fornitore)

Fornitore (fornitore, telefono_fornitore)

La tabella Magazzino non è in 3FN, perché non è in 2FN. Infatti, gli attributi indirizzo_azienda e telefono_azienda dipendono solo da una parte della chiave; per renderla in 2FN, è necessario dividerla in due tabelle diverse:

Magazzino (NumeroM, azienda, località, capacità_massima)

Azienda (azienda, indirizzo, telefono)

Il procedimento termina, perché sia la tabella Magazzino che Azienda sono in 3FN.

Lo schema finale in 3FN è il seguente:

Ordine (codiceO, quantità, prezzo, fornitore)

Fornitore (fornitore, telefono_fornitore)

Magazzino (NumeroM, azienda, località, capacità_massima)

Azienda (azienda, indirizzo, telefono)



FISSA LE CONOSCENZE

1. Quando una tabella si dice normalizzata?
2. Quando una relazione è in seconda forma normale?
3. Quando una tabella è in terza forma normale?
4. Che cos'è la dipendenza funzionale?

8. VINCOLI DI INTEGRITÀ REFERENZIALE

Nella Lezione 7 dell'Unità 1 abbiamo trattato le problematiche della sicurezza nelle basi di dati. Ora approfondiamo il tema dei vincoli di integrità con alcuni esempi.

Ricordiamo che un vincolo è una proprietà che deve essere soddisfatta da tutte le istanze corrette della base di dati e il DBMS stabilisce delle regole da rispettare per soddisfare tali vincoli. Queste regole possono essere semplici **vincoli intra-relazionali**, se sono definiti sugli attributi di una sola relazione (vincoli di unicità, vincoli di dominio e di tupla), limitando per esempio il dominio di esistenza di un campo, oppure i più sofisticati **vincoli inter-relazionali**, se sono definiti su più relazioni contemporaneamente, tenendo conto di interrelazioni fra i campi, anche di archivi diversi (**integrità referenziale**). I **vincoli di integrità** vengono definiti dall'amministratore della banca dati, che li codifica in uno Schema fornito a tutti i software applicativi che accedono alla base di dati.

Abbiamo visto nelle lezioni precedenti esempi di vincoli intra-relazionali. Soffermiamoci ora sui **vincoli di integrità referenziale**, analizzando alcuni esempi. Una tupla in Esami per essere valida deve contenere un numero di matricola che esista anche in Studenti. Lo studente di matricola 995566, presente in Esami, non è presente in Studenti; quindi è una tupla non valida.

Studenti

matricola	cognome	nome	nascita
254782	Rossi	Giacomo	21/10/1994
564387	Lorenzi	Paolo	20/09/1990
345218	Bianchi	Franca	10/12/1989
123459	Bruni	Chiara	17/07/1991
789657	Verdi	Luigi	20/06/1990

Esami

studente	voto	lode	corso
254782	30	X	A04
995566	28	-	A10
345218	25	-	A16
123459	30	X	A34
789657	30	-	A06

Nella [fig. 48](#) è mostrato un altro esempio di relazioni diverse correlate attraverso valori comuni di uno o più attributi. I valori assunti da un

attributo nella relazione referenziante devono esistere effettivamente come valori di un attributo nell'istanza della relazione referenziata.



fig. 48 Relazioni collegate attraverso le chiavi esterne

I vincoli di integrità referenziale vietano le operazioni che potrebbero rendere inconsistente la base di dati; in particolare sono rifiutati alcuni inserimenti e cancellazioni.

Per esempio, facendo riferimento alle tabelle Corsi e Docenti, si può notare che l'attributo MatrDocente nella relazione Corsi fa riferimento a matricola nella relazione Docenti. Perché sia rispettata l'integrità referenziale, è necessario che i valori assunti dall'attributo MatrDocente nella relazione Corsi esistano come valori dell'attributo matricola nella relazione Docenti.

Corsi

codice	nome	MatrDocente
M2170	Fondamenti di informatica	D101
M4880	Sistemi di elaborazione	D102
F0410	Basi di dati	D321

Docenti

matricola	nome	telefono
D101	Verdi	123456
D102	Bianchi	636363
D321	Neri	414243

fig. 49 Relazioni Corsi-Docenti

Se si impone un vincolo di integrità referenziale non è possibile effettuare un'operazione di inserimento di un Corso tenuto da un docente che non sia già presente nella tabella Docenti; inoltre non è possibile cancellare un docente se prima non sono stati cancellati tutti i corsi tenuti da quel docente.

FISSA LE CONOSCENZE

- Quali sono i vincoli intra-relazionali?
- Quando si impongono dei vincoli di integrità referenziale?
- Quali sono le operazioni vietate tra due tabelle con vincoli di integrità referenziale?



RIPASSIAMO INSIEME

I modelli logici

Esistono due tipi diversi di DBMS: il **modello a grafo**, basato su puntatori utilizzati come riferimenti esplicativi fra record di tipi diversi, e il **modello relazionale**. Ad oggi si stanno sviluppando anche i modelli Object-Oriented e i modelli NoRel, utilizzati soprattutto per gestire dati non strutturati.

Il modello relazionale

Nel modello relazionale, un database è un **insieme di tabelle**. Una tabella relazionale è formata da righe (tuple), che rappresentano le istanze degli oggetti, e da colonne, che rappresentano le proprietà degli oggetti. Le colonne assumono valori appartenenti a un dato dominio e possono contenere al massimo un valore. Solitamente vi è una colonna (o un insieme di colonne), detta **chiave primaria**, che identifica univocamente una riga della tabella.

Ristrutturazione dello schema E/R

Lo schema concettuale va **semplificato** per permettere una più agevole traduzione nel modello relazionale e una gestione più efficiente dei dati. In particolare vanno eliminate le gerarchie e gli attributi multipli, e si possono accoppare due tabelle o dividere una in due per ottimizzare gli accessi ai dati.

Traduzione nello schema logico

Per ottenere uno **schema logico relazionale** dallo schema concettuale E/R, le entità del diagramma E/R diventano tabelle e gli attributi i campi. La relazione di cardinalità 1:1 può essere realizzata introducendo in una delle due tabelle la chiave dell'altra. Nella relazione 1:N il legame fra le due tabelle viene creato aggiungendo agli attributi dell'entità "a molti" la chiave dell'entità "a uno".

Per risolvere invece la relazione M:N, si deve introdurre una nuova tabella contenente le chiavi delle due entità e gli eventuali attributi dell'associazione.

Operazioni nelle tabelle relazionali

Gli **operatori di base insiemistici** sono: unione, intersezione, differenza e prodotto cartesiano.

Gli **operatori di base relazionali** sono: selezione, che seleziona da una tabella le tuple che hanno una determinata caratteristica; proiezione, che prende solo le colonne richieste all'interno di ogni tupla della tabella; congiunzione, che seleziona i dati comuni da due o più tabelle tramite un operatore di confronto.

Algebra relazionale

L'algebra relazionale è un **linguaggio di interrogazione** ed è definita da operatori di base. L'applicazione di ognuno di questi operatori produce una relazione.

La normalizzazione

La normalizzazione di uno schema è un procedimento che ha come scopo quello di raffinare lo schema, eliminando eventuali **errori nell'organizzazione dei dati**. Esistono vari livelli di normalizzazione (forme normali). Una tabella è in 1FN quando tutti i suoi attributi sono definiti su domini elementari; è in 2FN quando tutti i suoi attributi dipendono dall'intera chiave; è in 3FN quando non vi è dipendenza transitiva.

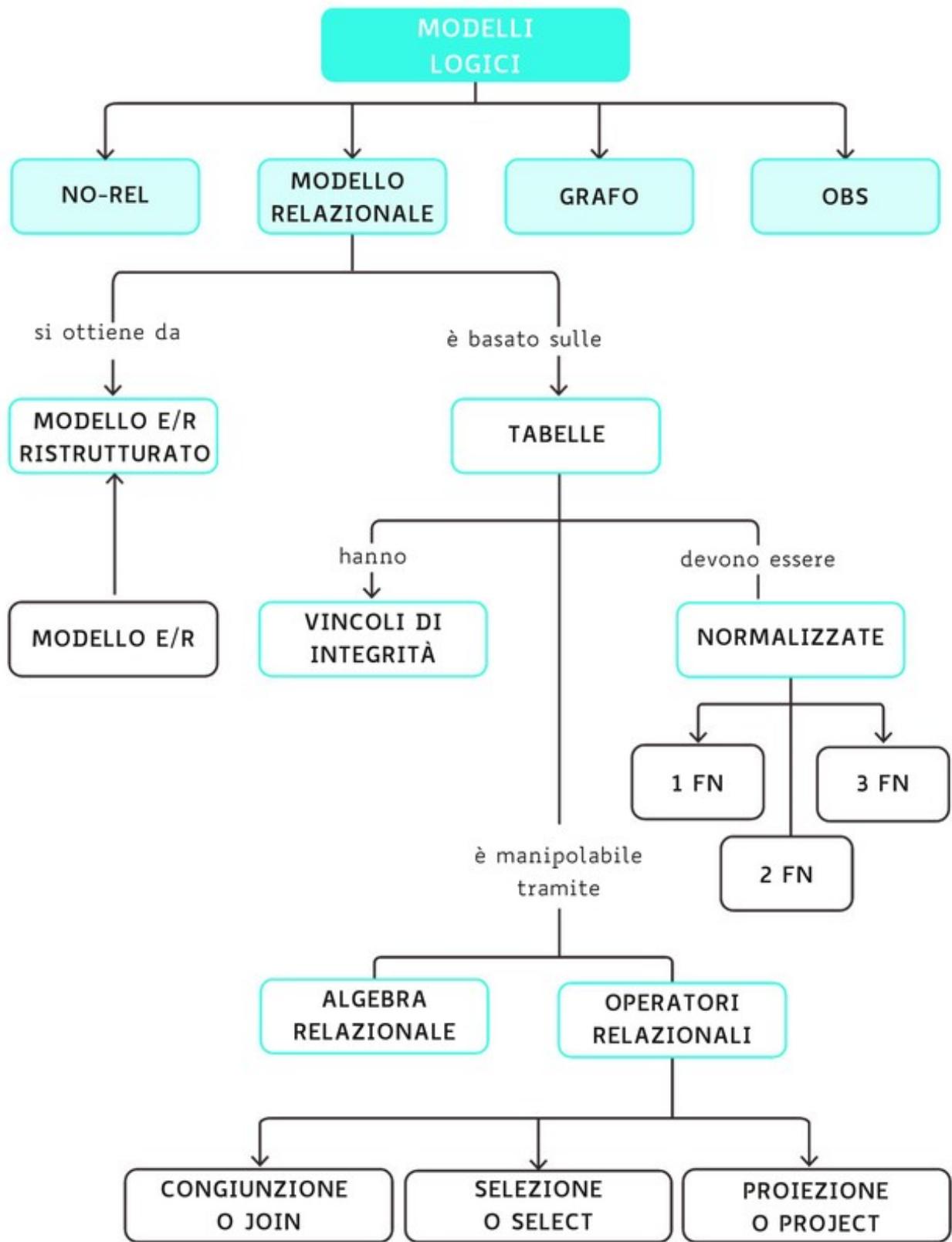
Vincoli di integrità

I vincoli di integrità sono vincoli che devono essere rispettati per garantire la **consistenza** dei dati all'interno delle basi di dati relazionali. Esistono vincoli intra-relazionali (di tupla, di dominio) e inter-relazionali (integrità referenziale).

MAPPA CONCETTUALE



MAPPA
MODIFICABILE



TEST**TEST**

Svolgi il test interattivo

Vero o falso?

1. Il modello relazionale è un modello concettuale.
2. Le righe di una tabella sono tutte diverse.

<input type="checkbox"/>	V
<input type="checkbox"/>	F

Scelta multipla (una sola è la risposta esatta)

3. Una tabella relazionale è composta da un insieme di:
 - A vettori.
 - B record.
 - C stringhe.
 - D numeri.
4. Il modello relazionale descrive:
 - A la singola vista.
 - B lo schema concettuale.
 - C lo schema logico.
 - D lo schema fisico.
5. La cardinalità di una tabella indica:
 - A il numero di righe.
 - B il numero di colonne.
 - C il numero di occorrenze.
 - D le ennuple.
6. Il grado di una tabella indica:
 - A il numero di attributi.
 - B il numero di righe.
 - C il numero di campi.
 - D la dimensione della tabella.
7. I vincoli di integrità referenziale servono per impedire:
 - A l'inserimento di nuovi dati.
 - B la cancellazione di dati non ancora validati.
 - C l'inconsistenza dei dati.
 - D la ridondanza dei dati.

PREPARATI PER IL COLLOQUIO ORALE

1. Come avviene la traduzione di un'associazione N:N dal modello E/R al modello relazionale? [vedi lez. 4]
2. Quale grado e quale cardinalità ha la tabella risultante di un'operazione di selezione? E di proiezione? E di congiunzione? [vedi lez. 5]
3. Che cosa indicano le prime tre forme normali? [vedi lez. 7]
4. Che cosa sono i vincoli di integrità referenziale? [vedi lez. 7]
5. Qual è la differenza tra vincoli intra-relazionali e inter-relazionali? [vedi lez. 7]
6. Quali sono gli operatori di base dell'algebra relazionale? [vedi lez. 5]
7. Come viene tradotta un'entità nel modello relazionale? [vedi lez. 4]
8. Quando una relazione è in 3FN? [vedi lez. 7]
9. Quale comando viene utilizzato per selezionare dei dati in una tabella di un database? [vedi lez. 5]
10. Quali sono le caratteristiche del modello a grafo? [vedi lez. 1]



CLIL – IN ENGLISH, PLEASE



AUDIO

Ascolta la pronuncia
del testo

ABSTRACT

Relational database

The relational model which until now has been most commonly used is based on the representation of data by means of tables.

The three relational operators

are selection which selects from a table the rows having a given characteristic, projection which takes only the requested columns within a table, and conjunction which selects the data from two or more columns when there are correspondences between the values of common attributes.

An identifier is one or more attributes that uniquely identify a row in a table.

One of the identifiers is selected as the primary key.

The degree of a table is the number of columns in that table, and the cardinality of a table is the number of rows.

The domain is the set of admissible values of a given attribute.

EXERCISES

True or false?

1. In the logical schema a one-to-many relationship (1:N) is implemented by introducing a new entity.
2. The degree of a table indicates the number of attributes.
3. The domain is the number of rows in a table.
4. The primary key uniquely identifies a row in a table.

Multiple choice

5. The projection yields:
 - A selected columns.
 - B selected rows.
 - C a table equivalent to the original.

GLOSSARY

Primary Key: is a key that uniquely defines the characteristics of each row (also known as record or tuple). The primary key has to consist of characteristics that cannot be duplicated by any other row.

Foreign Key: is a field (or collection of fields) in one table that uniquely identifies a row of another table. In other words, a foreign key is a column or a combination of columns that is used to establish and enforce a link between the data in two tables.

Relational Table: is a set of data elements (values) that is organized using a model of vertical columns (which are identified by their name) and horizontal rows, the cell being the unit where a row and column intersect.

Relational Algebra: is an offshoot of first-order logic and of algebra of sets concerned with operations over finitary relations, usually made more convenient to work with by identifying the components of a tuple by a name (called attribute) rather than by a numeric column index, which is called a relation in database terminology.

GLOSSARIO
CLIL

**COMPETENZE DISCIPLINARI**

- Identificare e applicare le metodologie e le tecniche della gestione per progetti.
- Redigere relazioni tecniche e documentare le attività individuali e di gruppo relative a situazioni professionali.
- Interpretare i sistemi aziendali nei loro modelli, processi e flussi informativi con riferimento alle differenti tipologie di imprese.

COMPETENZE DEL XXI SECOLO**Abilità fondamentali****CULTURA SCIENTIFICA**

Capacità di usare conoscenze scientifiche e regole/modelli per interpretare un fenomeno.

Competenze trasversali**PENSIERO CRITICO**

Saper analizzare e valutare situazioni in modo da impiegare informazioni e idee per formulare risposte e soluzioni.

COMUNICAZIONE

Saper ascoltare, comprendere e contestualizzare le informazioni, per poi trasmetterle ad altri (in modalità verbale o non verbale).

COLLABORAZIONE

Saper lavorare in gruppo in vista di un obiettivo comune, prevenendo e gestendo i conflitti.

Qualità caratteriali**CURIOSITÀ**

Inclinazione a porre domande con una mentalità aperta.

OBIETTIVI FORMATIVI	TEMPI	STRUMENTI

IL TEMA

Di seguito è riportato un estratto della prima parte della **prova di informatica** dell'Esame di Stato del 2015 per l'Istituto Tecnico, settore Tecnologico, indirizzo Informatica e Telecomunicazioni.

Da solo o in gruppo leggi il tema proposto e svolgilo.
In questa unità prendiamo in considerazione il punto 3, dove è chiesto di realizzare "uno schema logico della base di dati".

LA WEB COMMUNITY DEGLI EVENTI LIVE

Si vuole realizzare una web community per condividere dati e commenti relativi a eventi dal vivo di diverse categorie, ad esempio concerti, spettacoli teatrali, balletti, ecc. che si svolgono in Italia.

Gli eventi vengono inseriti sul sistema direttamente dai membri stessi della community, che si registrano sul sito fornendo un nickname, nome, cognome, indirizzo email e scegliendo una o più categorie di eventi a cui sono interessati.

Ogni membro iscritto riceve periodicamente per posta elettronica una newsletter, emessa automaticamente dal sistema, che riporta gli eventi delle categorie da lui scelte, che si svolgeranno nella settimana seguente nel territorio provinciale dell'utente.

I membri registrati possono interagire con la community sia inserendo i dati di un nuovo evento, per il quale occorre specificare categoria, luogo di svolgimento, data, titolo dell'evento e artisti coinvolti, sia scrivendo un post con un commento e un voto (da 1 a 5) su un evento. Il sito della community offre a tutti, sia membri registrati sia utenti

anonimi, la consultazione dei dati online, tra cui:

- visualizzazione degli eventi di un certo tipo in ordine cronologico, con possibilità di filtro per territorio di una specifica provincia;
- visualizzazione di tutti i commenti e voti relativi ad un evento.

Il candidato, fatte le opportune ipotesi aggiuntive, sviluppi:

[...]

3. uno schema logico della base di dati.

I nuclei tematici fondamentali e gli obiettivi

Fra i nuclei tematici fondamentali e gli obiettivi cui si riferiscono le prove d'esame (vedi la sezione finale "La preparazione alla seconda prova scritta dell'esame di Stato"), questo estratto si riferisce ai seguenti.

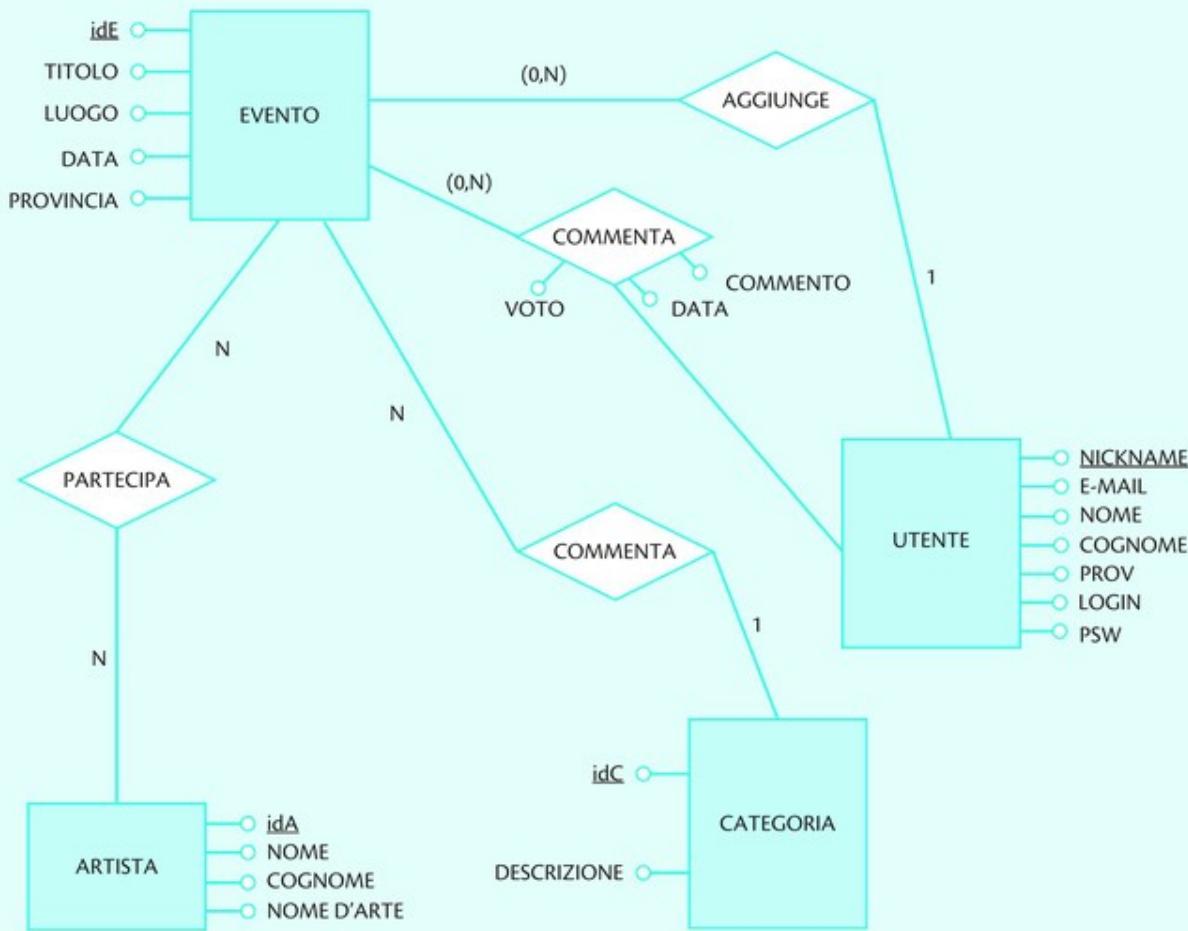
Nuclei tematici fondamentali	Obiettivi
<ul style="list-style-type: none"> • Progettazione di basi di dati: modellazione concettuale, logica e fisica di una base di dati. • Tecnologie per il Web: realizzazione di applicazioni web anche con interfacciamento a basi di dati. 	<ul style="list-style-type: none"> • Affrontare situazioni problematiche, utilizzando adeguate strategie cognitive e procedure operative orientate alla progettazione di soluzioni informatiche. • Scegliere sistemi e strumenti idonei al contesto proposto, in base alle loro caratteristiche funzionali. • Realizzare progetti secondo procedure consolidate e criteri di sicurezza.

Svolgimento

Dallo schema E/R progettato nell'Unità 2, che per comodità riportiamo di seguito, dobbiamo passare allo schema logico relazionale.

Lo schema E/R non ha bisogno di essere ristrutturato, perché non sono presenti gerarchie e non vi sono attributi multipli.





Per ottenere lo schema relazionale procediamo applicando le regole di derivazione.

Per ogni entità costruiamo una tabella che avrà gli stessi attributi dell'entità e come chiave primaria la stessa di quella dell'entità. Nello schema E/R sono presenti le associazioni Aggiunge e Appartiene, che sono di tipo 1:N. Per realizzare Aggiunge, facciamo migrare la chiave primaria dalla tabella Utente (nickname) nella tabella Evento, dove diventa chiave esterna (cod_utente). Facciamo lo stesso per Appartiene: la chiave primaria di categoria (idC) viene duplicata in Evento (cod_categoria). Per l' associazione Partecipa di tipo N:N costruiamo la tabella Partecipa, che avrà come attributi la chiave primaria di Artista e la chiave primaria di Evento. L'associazione Commenta è di tipo N:N ed ha tre attributi, per cui costruiamo la tabella Commenta, che avrà i tre attributi dell'associazione (voto, data, commento) e le chiavi primarie di Utente ed Evento.

Lo schema logico finale sarà il seguente:

Evento (idE, titolo, luogo, data, provincia, cod_utente, cod_categoria)

Artista (idA, nome, cognome, nome d'arte)

Categoria (idC, descrizione)

Utente (NICKNAME, e-mail, nome, cognome, prov, login, psw)

Partecipa (cod_artista, cod_evento)

Commenta (voto, data, commento, cod_evento, cod_utente)

Lo schema logico ottenuto risulta normalizzato poiché ciascuna relazione è in terza forma normale.

COMPITI DI REALTÀ

Dopo aver confrontato la risoluzione del tema d'esame con la tua, in gruppo svolgi tra le seguenti attività quelle che non hai già svolto nell'unità precedente.

- 1. COMPETENZA DIGITALE | COLLABORAZIONE** Effettuate una ricerca su Internet e analizzate lo **stato dell'arte** delle web *community* e dei *social network*: potete partire da questo link <https://tinyurl.com/y9vy-6svd>, in cui la piattaforma *We Are Social* fornisce un'analisi di tipo quantitativo, e da questo articolo di SkyTG24 <https://tinyurl.com/ydeb389p>, che individua alcune delle più importanti problematiche connesse con l'uso dei social. Potete usare anche altre fonti più recenti, ma ricordatevi di verificare sempre l'**attendibilità dei siti web che consultate**.
- 2. PENSIERO CRITICO | COLLABORAZIONE** Individuate quella che, secondo voi, è la **problematica più attuale** collegata all'uso dei social network: può essere la *privacy*, il tema delle *fake news*, o altri argomenti che vi sembrano attuali e importanti. Approfonditeli, discutendo in gruppo.
- 3. COMPETENZA DIGITALE | COLLABORAZIONE** Raccogliete i **risultati della vostra ricerca** e della vostra discussione in una **presentazione in PowerPoint** (o in un altro strumento di presentazione), formata al massimo da cinque slide.
- 4. COLLABORAZIONE | COMUNICAZIONE** In classe, con la supervisione dell'insegnante, condividete la presentazione con i compagni: confrontate le **tematiche emerse** e discutetele.
- 5. CULTURA SCIENTIFICA | COLLABORAZIONE** Infine, in classe, discutete con i compagni lo schema E/R presentato nella soluzione al tema d'esame e proponete **eventuali modifiche** che vi sembrano adatte a rappresentare, per **completezza e realismo**, la costruzione del database richiesto all'esame di Stato.

AUTOVALUTAZIONE

Al termine delle attività rifletti sull'esperienza e completa la tabella di autovalutazione.

Attività	LIVELLO			
	Iniziale	Base	Intermedio	Avanzato
Ho compreso senza difficoltà le richieste dell'attività proposta?	Ho compreso solo alcune richieste aiutato dal docente. <input type="checkbox"/>	Con la guida dell'insegnante e dei compagni ho compreso quasi tutte le richieste. <input type="checkbox"/>	Ho compreso le richieste ed in parte le ho svolte autonomamente. <input type="checkbox"/>	Ho identificato le richieste e le ho svolte senza difficoltà. <input type="checkbox"/>
Sono riuscito a progettare il database?	Ho impostato in modo minimale lo schema concettuale. <input type="checkbox"/>	Ho progettato il database in modo poco dettagliato e descrivendo gli elementi in modo sommario. <input type="checkbox"/>	Ho progettato il database tralasciando alcuni particolari. <input type="checkbox"/>	Ho realizzato uno schema concettuale completo e ben descritto. <input type="checkbox"/>
Sono in grado di usare gli strumenti informatici?	Ho descritto le entità ma non le associazioni. Non sono riuscito a leggere correttamente le associazioni. <input type="checkbox"/>	Ho documentato le entità e parzialmente le associazioni. <input type="checkbox"/>	Ho documentato le entità e le associazioni ma in modo incompleto. <input type="checkbox"/>	Ho documentato le entità e le associazioni in modo completo ed esaustivo. <input type="checkbox"/>

4

Il linguaggio SQL



ESERCIZI COMMENTATI

Segui la risoluzione passo passo



LABORATORI CASE STUDIES

Approfondisci con i casi pratici



PREREQUISITI

- Conoscere le caratteristiche del modello relazionale.
- Conoscere la programmazione procedurale.
- Conoscere gli aspetti base dell'analisi di un problema.

PER COMINCIARE

1. Una relazione è:
 - A una tabella suddivisa in più campi alfanumerici
 - B una struttura dati in grado di contenere valori numerici
 - C una tabella contenente nelle colonne gli attributi e nelle righe i valori
 - D una variabile strutturata che può contenere numeri o stringhe
2. Quale campo della tabella Studente (Matricola, Cognome, Nome, Età) può essere chiave primaria?
 - A Matricola
 - B Nome
 - C Cognome
 - D Età
3. Spiega come si realizzano le associazioni tra dati nel modello relazionale.
4. Definisci le relazioni che compongono il database Scuola con i dati degli studenti e degli insegnanti.



CONOSCENZE

- Conoscere le caratteristiche del linguaggio SQL e come utilizzarlo.
- Conoscere le principali istruzioni di DDL e di DML.
- Conoscere le principali istruzioni per la gestione delle viste e per la sicurezza dei dati.



ABILITÀ

- Saper utilizzare il linguaggio SQL.
- Saper definire lo schema.
- Saper costruire le query.
- Saper effettuare operazioni complesse.
- Saper garantire la sicurezza dei dati.

COMPETENZE

- Scegliere dispositivi e strumenti in base alle loro caratteristiche funzionali.
- Gestire progetti secondo le procedure e gli standard previsti dai sistemi aziendali di gestione della qualità e della sicurezza.

1. DEFINIRE LO SCHEMA

SQL (Structured Query Language) è un linguaggio che consente di inserire, ricercare, aggiornare, cancellare i dati memorizzati in una base di dati di tipo relazionale.

Come linguaggio per le basi di dati relazionali, SQL fa parte dei **linguaggi non procedurali**. Per "non procedurale" si intende che gli utenti devono soltanto specificare quali dati desiderano e non come individuarli. Java e C sono, invece, esempi di linguaggi procedurali.

SQL può essere utilizzato sia in modo interattivo (quando l'utente digita il comando, quest'ultimo viene immediatamente eseguito) sia in modo *embedded*, cioè all'interno di programmi scritti con linguaggi tradizionali (per esempio C#). I risultati delle istruzioni in questo caso non sono immediatamente visibili all'utente, ma vengono elaborati dal programma ospite. SQL nasce come prototipo a metà degli anni Settanta del Novecento (SEQUEL), ma solo negli anni Ottanta si afferma come linguaggio per il software dei database relazionali (DB2 e SQL/DS). Nel 1986 l'*ANSI (American National Standards Institute)*, cioè l'associazione americana che si occupa di definire gli standard nazionali, adottò SQL come standard per i linguaggi relazionali; nel 1987 SQL divenne anche standard ISO. Il linguaggio SQL è stato implementato da numerosi produttori e ovviamente esistono differenze tra un'implementazione e l'altra, ma tutte sono comunque aderenti agli standard internazionali. Attualmente, tra i prodotti più comuni che implementano SQL, vi sono Oracle, Informix, Microsoft Access, SQL Server, MySQL.

Il linguaggio SQL è progettato per:

- creare e modificare schemi di database (*DDL - Data Definition Language*);
- inserire, modificare e gestire dati memorizzati (*DML - Data Manipulation Language*);
- interrogare dati memorizzati (*DQL - Data Query Language*);
- creare e gestire strumenti di controllo e accesso ai dati (*DCL - Data Control Language*);
- creare e gestire le transazioni (*TCL - Transaction Control Language*).

■ Creazione di database

Per creare un database si usa la seguente sintassi:

```
CREATE DATABASE database_name
```

Volendo fare un esempio pratico, per creare il database per la gestione delle auto di una concessionaria possiamo digitare:

```
CREATE DATABASE dbAuto
```

Per eliminare un database esistente si usa il comando:

```
DROP DATABASE database_name
```

LO SAI CHE

Utilizzando linguaggi di programmazione (Java, C#, PHP, ASPX) si possono scrivere programmi che popolano una tabella, prelevando i dati memorizzati in un file di testo.

■ Creazione di tabelle

Create table

Per creare una tabella in cui registrare dati si usa l'istruzione CREATE TABLE:

```
CREATE TABLE nome-tabella  
  (nome-colonna1 tipo1 {NOT NULL}  
  .....  
  nome-colonnaN tipoN)
```

nome-colonna1 e nome-colonnaN sono i **nomi delle colonne** della tabella. Accanto a ogni nome di colonna deve essere specificato il **tipo di dati** e, se necessario, altre **clausole**.

Le principali clausole che possono essere definite sono:

- PRIMARY KEY definisce il campo come chiave primaria (univoca);
- FOREIGN KEY definisce il campo come chiave straniera (esterna);
- UNIQUE indica che il campo non può avere valori duplicati;
- NOT NULL indica che il campo deve avere sempre un valore.

Tipi di dati

In SQL possono essere definiti dati numerici (interi e reali), alfanumerici (stringhe e date) e oggetti. Esistono poi tipi standard per l'utilizzo.

Ogni implementazione di SQL, pur nel rispetto delle norme generali, introduce piccole variazioni o nuovi tipi. Vediamo quelli più utilizzati e normalmente validi per quasi tutti i sistemi, ricordando che alcuni tipi possono essere abbreviati nella loro definizione (per esempio CHAR – CHARACTER, INT – INTEGER).

Tipo	Nome	Contenuto
Numerico	SMALLINT	Numero intero tra -32768 e 32767 (2 byte)
	INT	Numero intero tra -2147483648 e 2147483647 (4 byte)
	FLOAT	Numero reale in singola precisione (4 byte)
	DOUBLE	Numero reale in doppia precisione (4 byte)
	DECIMAL(p,q)	Numero con un massimo di p cifre di cui q cifre dopo il punto decimale. Occupa una dimensione che può variare da 5 a 17 byte a seconda del valore di p
Logico	BOOLEAN	TRUE , FALSE
Testo	CHAR(n)	Stringa di massimo n caratteri (dove n è maggiore di 0 e minore di 255); ogni valore della colonna richiede n caratteri
	VARCHAR(n)	Stringa di massimo n caratteri (dove n è maggiore di 0 e minore di 2048 secondo il prodotto SQL); ogni valore della colonna richiede uno spazio pari al numero di caratteri effettivi del campo
	MEMO	Stringa fino a 65536 caratteri
Data	DATE	Data nella forma AAAA/MM/GG
	TIME	Ora nella forma HH:MM

Si deve comunque tener presente che i tipi di dati e le relative occupazioni di memoria possono variare da un DBMS all'altro.

Nell'esempio è mostrata l'istruzione per creare una tabella relativa ad alcuni giocatori.

```
CREATE TABLE Giocatori
(Id_G          INT PRIMARY KEY,
 Cognome       VARCHAR(15)    NOT NULL,
 Sesso         CHAR(1)        NOT NULL,
 Indirizzo     VARCHAR(25),
 Punti         INT,
 Cod_Squadra   CHAR(3),
 FOREIGN KEY(Cod_Squadra) REFERENCES Squadra(Id_Squadra))
```

Ogni colonna NOT NULL deve riportare un valore in ogni riga; quindi ogni giocatore deve avere un Cognome, ma non necessariamente un Indirizzo. SQL accetta NULL come possibile valore per una colonna.

Tale valore può essere paragonato a "valore sconosciuto" oppure "valore non specificato" e non deve essere confuso con il numero zero o con gli spazi.

La clausola FOREIGN KEY indica che il campo Cod_Squadra è una chiave esterna e si riferisce alla chiave primaria Id_Squadra della tabella Squadra (Squadra(Id_Squadra)). Per definire la chiave esterna alcuni DBMS accettano anche la seguente sintassi:

```
Cod_Squadra   CHAR(3) REFERENCES Squadra(Id_Squadra)
```

I vincoli di integrità referenziale consentono di definire le operazioni eseguite da SQL quando un utente tenta di eliminare o aggiornare una chiave alla quale fa riferimento una chiave esterna.

Le clausole REFERENCES di CREATE TABLE supportano le direttive:

- [ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }];
 - [ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }].
- ON DELETE NO ACTION (è il valore predefinito): specifica che se si tenta di eliminare una riga contenente una chiave a cui fanno riferimento chiavi esterne presenti in righe in altre tabelle, verrà generato un errore e la riga non verrà eliminata.
 - ON UPDATE NO ACTION: specifica che se si tenta di aggiornare un valore di chiave in una riga e alla chiave fanno riferimento chiavi esterne in righe esistenti in altre tabelle, verrà generato un errore e verrà annullata la modifica apportata con UPDATE.
 - ON DELETE CASCADE: specifica che se si tenta di eliminare una riga contenente una chiave a cui fanno riferimento chiavi esterne in righe esistenti in altre tabelle, verranno eliminate anche tutte le righe contenenti tali chiavi esterne.
 - ON UPDATE CASCADE: specifica che se si tenta di aggiornare un valore di chiave in una riga e a tale valore fanno riferimento chiavi esterne in righe esistenti in altre tabelle, tutti i valori che compongono la chiave esterna verranno anch'essi aggiornati al nuovo valore specificato per la chiave.

È possibile fornire a una colonna un valore predefinito che sarà utilizzato quando, al momento di inserire i dati nella tabella, non verrà specificato un valore con la clausola DEFAULT "valore" dopo la dichiarazione del tipo di dati.

Nella tabella Giocatori dell'esempio precedente assegniamo il valore predefinito "M" al campo sesso con la seguente sintassi:

```
Sesso CHAR(1) default "M";
```

Inoltre, per verificare che il valore da inserire nel campo rientri in un certo range o abbia un determinato formato, si utilizza la clausola CHECK. Si scrive:

```
punti int CHECK (punti >= 0),
```

Per il campo punti saranno ritenuti validi valori maggiori o uguali a 0.

■ Creare un indice

Per reperire uno o più record in una tabella, SQL normalmente effettua un **accesso sequenziale alla tabella** leggendo una riga per volta. Se si cerca una sola riga e la tabella è composta da numerose righe, questo metodo richiede ovviamente molto tempo ed è quindi poco efficiente.

Per velocizzare la ricerca è necessario effettuare un **accesso diretto ai dati**, ma per poterlo fare bisogna definire un **indice** sul campo di ricerca.

L'istruzione CREATE INDEX serve per creare un indice su una tabella esistente. La sua sintassi è la seguente:

```
CREATE UNIQUE INDEX nome-indice  
ON nome-tabella(nome-colonna)
```

L'uso degli indici è molto importante per rendere **più veloci le operazioni di interrogazione della base di dati** anche se, durante l'elaborazione, è SQL a stabilire se utilizzare o meno uno degli indici presenti.

Quando si aggiornano, si inseriscono o si cancellano righe di una tabella, SQL deve **rivedere gli indici** definiti sulla stessa. Ciò significa che, se da un lato le operazioni di ricerca sono più veloci, dall'altro il tempo di elaborazione delle istruzioni di modifica dei dati è maggiore.

L'opzione UNIQUE specifica che non si desiderano valori duplicati per l'indice; quando è omessa, i duplicati sono ammessi.

È possibile definire più di un indice sulla stessa tabella. Se si prevede di effettuare frequentemente ricerche utilizzando due o più campi alla volta, è possibile creare un indice per quella **combinazione di campi** (indice multi-colonna) inserendo i nomi dei campi separati dalla virgola all'interno della clausola ON:

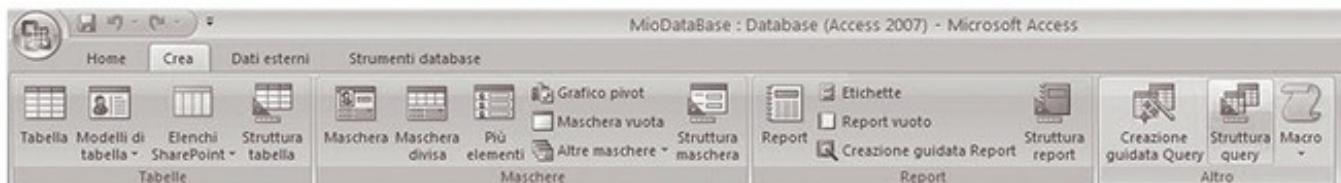
```
CREATE UNIQUE INDEX nome-indice  
ON nome-tabella(nome-colonna1, nome-colonna2)
```

Come utilizzare SQL

Per scrivere le query SQL possiamo utilizzare Microsoft Access nel seguente modo:

- 1 nella finestra principale di Microsoft Access selezionare da menu Crea e scegliere Struttura query (fig. 1);

fig. 1



- 2 dalla finestra Mostra tabella non inserire alcuna tabella, ma selezionare Chiudi (fig. 2);

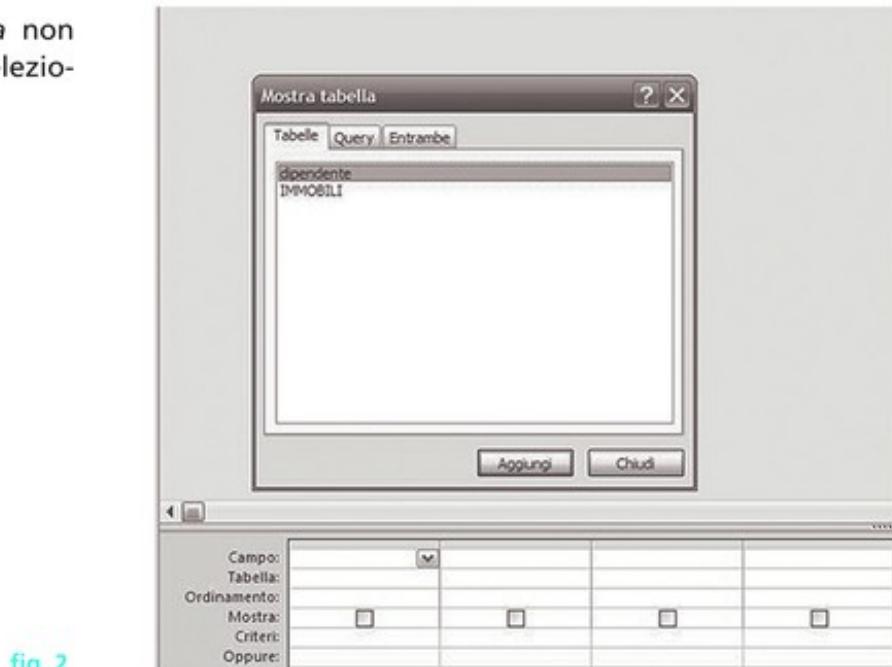


fig. 2

- 3 dal menu scegliere Home e poi SQL (fig. 3);

fig. 3



- 4 si apre una finestra dove è possibile scrivere il comando SQL (fig. 4). Una volta terminato è possibile salvare la query ed eseguirla come una normale query Access.



fig. 4

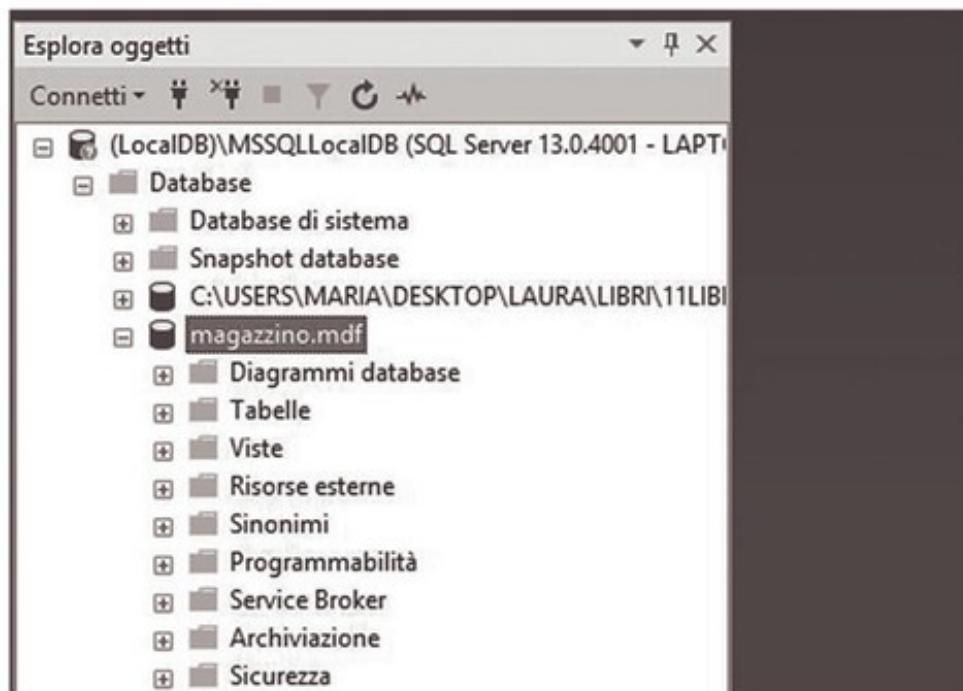
Per scrivere le query in SQL Server sono disponibili diversi strumenti, tra cui:

- SQL Operations Studio;
- SQL Server Management Studio (SSMS);
- MSSQL-cli, che è uno strumento da riga di comando interattivo per le query di SQL Server.

Vediamo come si usa SQL con SQL Server Management Studio, che dispone di una buona interfaccia utente (GUI):

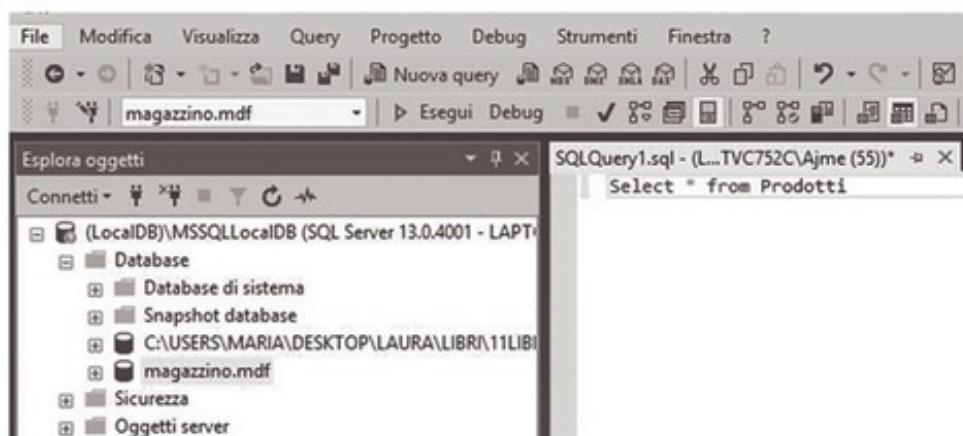
- 1 dopo aver selezionato il nome del server nella pagina iniziale di SSMS (che in Microsoft SQL Server 2017 è (LocalDB)\MSSQLLocalDB), compare una videata in cui è possibile scegliere il nome del database su cui lavorare (fig. 5):

fig. 5



- 2 selezionare il pulsante *Nuova query* per scrivere il comando SQL nella parte di destra della finestra;
- 3 selezionare il pulsante *Esegui* per eseguire la query (fig. 6).

fig. 6



Per scrivere le query in MySQL possiamo utilizzare PhpMyAdmin:

- 1 dopo aver selezionato il database dalla pagina iniziale di PhpMyAdmin, selezionare il pulsante *SQL* (fig. 7);

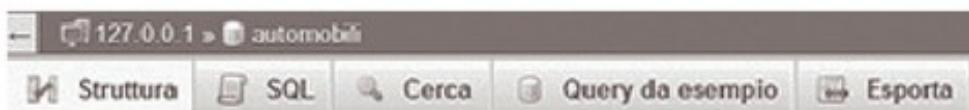


fig. 7

- 2 scrivere il comando SQL;
- 3 selezionare il pulsante *Esegui*, per eseguire la query (fig. 8).

fig. 8

■ La base di dati per gli esercizi guidati

Negli esercizi che seguono, faremo riferimento alla base di dati DbAuto, ideata per gestire le automobili vendute. Il suo schema logico è:

- auto (targa, cilindrata, prezzo, modello, colore, alimentazione);
- modello (nomeModello, marca);
- marca (nomeMarca, sede, nazione).

targa	cilindrata	prezzo	modello	colore	alimentazione
AB001LX	1600	14000	Xsara	grigio	diesel
AB342CH	999	9000	Panda	rosso	benzina
BA666LL	1350	45000	33	blu	benzina
CB401ET	2000	15000	Punto	blu	diesel
CB453ER	1250	13000	Micra	bianco	benzina
CF786JY	1350	11000	Clio	verde	diesel
DH306HD	1350	13000	Punto	rosso	diesel
EF786JY	1600	14000	Clio	nero	benzina

Modello

nomeModello	marca
33	Alfa Romeo
Clio	Renault
Micra	Nissan
Panda	Fiat
Punto	Fiat
Xsara	Citroën
600	Fiat
Mito	Alfa Romeo

Marca

nomeMarca	sede	nazione
Alfa Romeo	Milano	Italia
BMW	Monaco	Germania
Citroën	Parigi	Francia
Fiat	Torino	Italia
Nissan	Tokyo	Giappone
Renault	Parigi	Francia

LABORATORIO

IL PROBLEMA Crea la tabella Auto che ha la seguente struttura:

Nome	Descrizione	Tipo	Lunghezza	Chiave
Targa	numero di targa	stringa	8 byte	primaria
Cilindrata	cilindrata	numerico	2 byte	
Prezzo	prezzo in euro	numerico	4 byte	
Modello	modello	stringa	20 byte	
Colore	colore	stringa	20 byte	
Alimentazione	tipo di alimentazione (diesel, benzina...)	stringa	10 byte	

Dopo averla creata, definisci:

- un indice sul campo Modello;
- un indice multi-colonna sui campi Marca e Colore.

L'ANALISI Per prima cosa occorre creare una tabella inserendo i campi previsti e definendo, oltre a una chiave primaria, due chiavi esterne.

LA CODIFICA SQL

```
CREATE TABLE Auto
(targa          VARCHAR (8) NOT NULL UNIQUE PRIMARY KEY,
cilindrata     INT,
prezzo         FLOAT,
modello        VARCHAR(20),
colore         VARCHAR(20),
alimentazione  VARCHAR(10),
FOREIGN KEY (modello) REFERENCES Modello(nomeModello))
```

oppure:

```
CREATE TABLE Auto
(targa          VARCHAR (8) NOT NULL UNIQUE PRIMARY KEY,
cilindrata     INT,
prezzo         FLOAT,
modello        VARCHAR(20) REFERENCES Modello(nomeModello),
colore         VARCHAR(20),
alimentazione  VARCHAR(10))
```

IL RISULTATO

targa	cilindrata	prezzo	modello	colore	alimentazione

Dopodiché occorrerà definire un indice sul campo Modello della tabella Auto tramite il comando CREATE INDEX. Poiché potranno esistere più auto con lo stesso modello, non verrà utilizzata la clausola UNIQUE.

LA CODIFICA SQL

```
CREATE INDEX IndMod
Auto[modello]
```

Infine bisognerà definire l'indice multi-colonna sempre usando l'istruzione CREATE INDEX, ma utilizzando i campi Marca e Colore.

Poiché potranno esistere più auto con la stessa marca e lo stesso colore, non verrà utilizzata la clausola UNIQUE.

LA CODIFICA SQL

```
CREATE INDEX IndMod
ON Auto[modello,colore]
```

FISSA LE CONOSCENZE

1. Che cosa indica l'acronimo SQL?
2. Che differenza c'è tra un linguaggio procedurale e uno non procedurale?
3. Che effetto ha la clausola NOT NULL su una colonna?
4. A che cosa si riferisce la clausola REFERENCES di CREATE TABLE?
5. Come si definisce una chiave esterna?
6. A che cosa servono gli indici sui campi di ricerca?

2. MODIFICARE LO SCHEMA DI UNA BASE DATI

Lo schema dei dati può essere modificato sia eliminando tabelle o indici, sia modificando la struttura stessa della tabella.

■ Modificare la struttura di una tabella

L'istruzione ALTER TABLE permette all'utente di modificare la struttura di una tabella. La clausola successiva indica il tipo di modifica che si vuole effettuare sulla tabella (aggiunta, eliminazione e modifica).

Per inserire una colonna in una tabella già creata, si usa:

```
ALTER TABLE identificatore-tabella  
ADD nome-colonna tipo
```

Per esempio, il comando:

```
ALTER TABLE Giocatori  
ADD Nazione CHAR(6)
```

inserisce il campo Nazione nella tabella Giocatori.

Per cancellare una colonna da una tabella già esistente, si usa:

```
ALTER TABLE identificatore-tabella  
DROP nome-colonna
```

Per esempio, il comando:

```
ALTER TABLE Giocatori  
DROP Nazione
```

cancella il campo Nazione nella tabella Giocatori.

Per modificare una colonna cambiandone il tipo, si usa:

```
ALTER TABLE identificatore-tabella  
CHANGE nome-colonna nome-colonna nuovotipo
```

Per esempio:

```
ALTER TABLE Giocatori  
CHANGE Punti Punti DECIMAL(3,2)
```

■ Eliminare tabelle e indici

Per cancellare una tabella si usa il comando:

```
DROP TABLE identificatore-tabella
```

Il comando DROP elimina sia la struttura della tabella sia tutte le righe in essa contenute.

Per cancellare un indice definito su una tabella, si usa:

```
DROP INDEX nome-indice ON identificatore-tabella
```

LABORATORIO

IL PROBLEMA Inserisci il nuovo campo km (kilometri percorsi dall'auto) nella tabella Auto descritta nella Lezione 1.

L'ANALISI Occorre modificare la tabella utilizzando l'istruzione ALTER TABLE e indicando il nuovo campo da inserire (km).

LA CODIFICA SQL

```
ALTER TABLE Auto  
ADD km INT
```

IL RISULTATO

targa	cilindrata	prezzo	modello	colore	alimentazione	km

LABORATORIO

IL PROBLEMA Cancella dalla tabella Auto il campo km.

L'ANALISI Occorre modificare la tabella utilizzando l'istruzione ALTER TABLE e indicando il campo da eliminare (km).

LA CODIFICA SQL

```
ALTER TABLE Auto  
DROP km
```

IL RISULTATO

targa	cilindrata	prezzo	modello	colore	alimentazione

FISSA LE CONOSCENZE

1. Quale comando si usa per aggiungere un campo in una tabella?
2. Quale comando si usa per eliminare una tabella?
3. Quale comando si usa per eliminare un campo di una tabella?
4. Quale comando si usa per modificare la lunghezza di un campo in una tabella?
5. Quale comando si usa per eliminare un indice?

3. MODIFICARE I DATI

Nelle lezioni precedenti abbiamo esaminato i comandi SQL che implementano le funzioni di DDL (*Data Definition Language*) ovvero quelle istruzioni che permettono di progettare, modificare e distruggere le strutture che contengono i dati. Esaminiamo ora la parte di SQL che assolve alle funzioni del DML (*Data Manipulation Language*) per inserire, modificare e cancellare i dati che sono memorizzati nelle tabelle.

■ Inserimento

È possibile inserire nuove righe in una tabella usando il comando:

```
INSERT INTO identificatore-tabella  
VALUES([costante1, costante2...])
```

Per esempio, per inserire una nuova riga nella tabella Giocatori, definita nella Lezione 1, si scrive:

```
INSERT INTO Giocatori  
VALUES (10, 'Rossi', 'M', 'Via Torino 10', 4, 'Sq1')
```

Se si vuole inserire una riga nella tabella riga, ma non tutti i campi hanno un valore, bisogna specificare i nomi dei campi di cui si conosce il valore:

```
INSERT INTO identificatore-tabella  
(nome-campoX, nomecampoY...) VALUES (valoreX, valoreY...)
```

Se, per esempio, si vuole inserire un nuovo giocatore del quale si conoscono solo il codice, il cognome e il codice della squadra in cui gioca, occorre scrivere:

```
INSERT INTO Giocatori  
(idG, Cognome, CodSquadra) VALUES (14, 'Bruni', 'sq3')
```

Alcuni ambienti, come per esempio MySQL e SQL Server, danno la possibilità di inserimenti multipli, vale a dire di inserire più record utilizzando un solo comando INSERT:

```
INSERT INTO Giocatori  
(idG, cognome, sesso, indirizzo, punti, codSquadra)  
VALUES  
(12, 'Verdi', 'F', 'Via Genova 22', 15, sq2),  
(15, 'Bianchi', 'M', 'Via Roma14', 5, sq1)
```

■ Modifica

Per modificare i valori contenuti in una tabella, si usa l'istruzione UPDATE:

```
UPDATE identificatore-tabella  
SET nome-colonna = espressione  
{WHERE condizione}
```

Quando è specificata la clausola WHERE, la modifica interessa solo quelle righe della tabella che soddisfano il criterio di selezione. In sua assenza, la modifica avrà effetto su tutte le righe della tabella.

Per modificare l'indirizzo del giocatore Rossi si ha:

```
UPDATE Giocatori  
SET Indirizzo = 'Via Milano 20'  
WHERE Cognome = 'Rossi'
```

Per incrementare di 1 il punteggio di tutti i giocatori, si scrive:

```
UPDATE Giocatori  
SET Punti = Punti + 1
```

■ Cancellazione

È possibile eliminare le righe di una tabella con l'istruzione DELETE:

```
DELETE  
FROM identificatore-tabella  
{WHERE condizione}
```

Anche in questo caso la presenza della clausola WHERE permette di eliminare solo i record che rispettano certe condizioni. L'assenza della clausola WHERE provoca lo svuotamento (non la cancellazione) di tutta la tabella.

Per cancellare le informazioni relative al giocatore Rossi, si scrive:

```
DELETE  
FROM Giocatori  
WHERE Cognome = 'Rossi'
```

LABORATORIO

IL PROBLEMA Inserisci i dati di una nuova automobile nella tabella Auto.

L'ANALISI Occorre ricordare che i dati saranno inseriti nei campi della tabella Auto nell'ordine in cui si presentano nella clausola VALUES. I dati alfanumerici devono inoltre essere racchiusi tra apici.

LA CODIFICA SQL

```
INSERT INTO Auto  
(targa, cilindrata, prezzo, modello, colore, alimentazione)  
VALUES ('CD564BA', 1200, 11000, 'Punto', 'bianco', 'benzina')
```

Al termine dell'esecuzione di questo comando non viene mostrato nulla. Per verificare il corretto inserimento dei dati, bisogna aprire la tabella Auto.

IL RISULTATO

targa	cilindrata	prezzo	modello	colore	alimentazione
AB001LX	1600	14000	Xsara	grigio	diesel
AB342CH	999	9000	Panda	rosso	benzina
BA666LL	1350	45000	33	blu	benzina
CB401ET	2000	15000	Punto	blu	diesel
CB453ER	1250	13000	Micra	bianco	benzina
CF786JY	1350	11000	Clio	verde	diesel
CD564BA	1200	11000	Punto	bianco	benzina
DH306HD	1350	13000	Punto	rosso	diesel
EF786JY	1600	14000	Clio	nero	benzina

LABORATORIO

IL PROBLEMA Modifica il colore dell'auto con targa CB453ER.

L'ANALISI Setta a rosso il campo colore del record in cui Targa = 'CB453ER'.

LA CODIFICA SQL

```
UPDATE Auto  
SET colore = 'rosso'  
WHERE targa = 'CB453ER'
```

IL RISULTATO

targa	cilindrata	prezzo	modello	colore	alimentazione
AB001LX	1600	14000	Xsara	grigio	diesel
AB342CH	999	9000	Panda	rosso	benzina
BA666LL	1350	45000	33	blu	benzina
CB401ET	2000	15000	Punto	blu	diesel
CB453ER	1250	13000	Micra	rosso	benzina
CF786JY	1350	11000	Clio	verde	diesel
CD564BA	1200	11000	Punto	bianco	benzina
DH306HD	1350	13000	Punto	rosso	diesel
EF786JY	1600	14000	Clio	nero	benzina

LABORATORIO

IL PROBLEMA Incrementa del 10% il prezzo delle auto Punto.

L'ANALISI Individua il modello e poi effettua un calcolo sul campo prezzo aggiornandolo.

LA CODIFICA SQL

```
UPDATE Auto  
SET prezzo = prezzo * 1.1  
WHERE modello='Punto'
```

IL RISULTATO

targa	cilindrata	prezzo	modello	colore	alimentazione
AB342CH	999	9000	Panda	rosso	benzina
BA666LL	1350	45000	33	blu	benzina
CB401ET	2000	16500	Punto	blu	diesel
CB453ER	1250	13000	Micra	rosso	benzina
DH306HD	1350	14300	Punto	rosso	diesel
EF786JY	1600	14000	Clio	nero	benzina

* alcune righe sono state omesse

LABORATORIO

IL PROBLEMA Cancella l'auto con targa AB001LX.

L'ANALISI Cancella dalla tabella Auto il record relativo all'auto in cui Targa = 'AB001LX'.

LA CODIFICA SQL

```
DELETE *  
FROM AUTO  
WHERE Targa = 'AB001LX'
```

Viene chiesta conferma dell'intenzione di cancellare un record. Al termine dell'esecuzione di questo comando non viene mostrato nulla; per verificare la cancellazione, bisogna aprire la tabella Auto e si noterà che non c'è più l'auto con targa AB001LX.

IL RISULTATO

targa	cilindrata	prezzo	modello	colore	alimentazione
AB342CH	999	9000	Panda	rosso	benzina
BA666LL	1350	45000	33	blu	benzina
CB401ET	2000	22000	Punto	blu	diesel
CB453ER	1250	13000	Micra	rosso	benzina
CF786JY	1350	16500	Clio	verde	diesel
CD564BA	1200	11000	Punto	rosso	benzina
DH306HD	1350	30000	Punto	rosso	diesel
EF786JY	1600	15400	Clio	nero	benzina

4. L'ISTRUZIONE SELECT

Il DQL (*Data Query Language* – linguaggio di interrogazione dei dati) comprende i comandi per leggere ed elaborare i dati presenti in un database strutturato secondo il modello relazionale.

L'istruzione SQL che consente di interrogare la base di dati è **SELECT**: il suo risultato è una tabella che successivamente può essere ancora elaborata. Il formato completo dell'istruzione **SELECT** è:

```
SELECT identificatore-colonna  
FROM identificatore-tabella  
{WHERE condizione}  
{GROUP BY [nome-colonna]...}  
{HAVING condizione}  
{ORDER BY [identificatore-colonna]}...
```

Questa istruzione permette di selezionare, dalla tabella specificata nella clausola **FROM**, tutte le righe che soddisfano la condizione specificata nella clausola **WHERE**. Il significato delle altre opzioni (**GROUP BY**, **HAVING**, **ORDER BY**) sarà analizzato in seguito.

Per esempio, per selezionare due campi (**Cognome** e **Indirizzo**) appartenenti alla tabella **Giocatori**, l'istruzione è la seguente:

```
SELECT Cognome, Indirizzo  
FROM Giocatori
```

L'utilizzo del carattere ***** indica la selezione di tutti i campi. Con l'istruzione:

```
SELECT *  
FROM Giocatori
```

vengono selezionati tutti i campi appartenenti alla tabella **Giocatori**.

LABORATORIO

IL PROBLEMA Visualizza la targa, il modello e il prezzo di tutte le auto.

L'ANALISI Nel comando **SELECT** inserisci il nome delle colonne che vuoi visualizzare (**operazione di proiezione**).

LA CODIFICA SQL

```
SELECT targa, modello, prezzo  
FROM Auto
```

IL RISULTATO

targa	modello	prezzo
AB001LX	Xsara	14000
AB342CH	Panda	9000
BA666LL	33	45000
CB401ET	Punto	15000
CB453ER	Micra	13000
CF786JY	Clio	11000
DH306HD	Punto	14300
EF786JY	Clio	14000

■ Clausola WHERE

La clausola WHERE definisce la **condizione** a cui devono sottostare le righe da ricercare. Una condizione può essere di confronto semplice (sono utilizzabili i classici operatori `=`, `<`, `>`, `>=`, `<=`, `<>`) o di confronto composto quando sono presenti gli operatori AND, OR, NOT.

```
SELECT Cognome, Indirizzo  
FROM Giocatori  
WHERE Sesso = 'M'
```

In questo esempio viene selezionato il cognome e l'indirizzo dei maschi (Sesso = 'M') presenti nella tabella Giocatori.

LABORATORIO

IL PROBLEMA Visualizza tutte le auto con prezzo minore di 20.000 euro.

L'ANALISI Nel comando SELECT imposta la condizione WHERE Prezzo < 20000 che specifica il criterio di selezione (**operazione di selezione**).

LA CODIFICA SQL

```
SELECT *  
FROM Auto  
WHERE prezzo < 20000
```

IL RISULTATO

targa	cilindrata	prezzo	modello	colore	alimentazione
AB001LX	1600	14000	Xsara	grigio	diesel
AB342CH	999	9000	Panda	rosso	benzina
CB401ET	2000	15000	Punto	blu	diesel
DH306HD	1350	14300	Punto	rosso	diesel
EF786JY	1600	14000	Clio	nero	benzina
CF786JY	1350	11000	Clio	verde	diesel
CB453ER	1250	13000	Micra	bianco	benzina

Il **predicato** della clausola WHERE può essere anche:

LIKE	: è similare al valore
NOT LIKE	: non è similare al valore
BETWEEN	: è compreso tra i due valori specificati
NOT BETWEEN	: non è compreso tra i due valori specificati
IS NULL	: è il valore null
IS NOT NULL	: non è il valore null
IN	: corrisponde a uno dei valori specificati
NOT IN	: non corrisponde ad alcuno dei valori specificati

LIKE

L'operatore LIKE permette un **confronto con stringhe** che contengono valori speciali, detti **caratteri jolly**, che sono i simboli percento '%' e underscore '_':

- '_' (underscore) per indicare un qualsiasi carattere singolo in quella posizione della stringa;
- '%' (percento) per indicare una sequenza qualsiasi di caratteri in quella posizione della stringa.

N.B.: in Access, i caratteri '%' e '_' sono sostituiti rispettivamente da '*' e '?'. Per esempio, se si vogliono ottenere tutti gli elementi che iniziano per una certa lettera si utilizza l'operatore LIKE come segue:

```
SELECT Nome_prodotto, Prezzo  
FROM Prodotti  
WHERE Nome_prodotto LIKE 'P%'
```

Il risultato è la visualizzazione dei campi Nome_prodotto e Prezzo, dei record appartenenti alla tabella Prodotti, ma solo quelli in cui Nome_prodotto inizia con la lettera P seguita da un qualunque insieme di caratteri (in Access il carattere '%' dovrebbe essere sostituito da '*').

Vediamo un altro esempio. La query:

```
SELECT Nome, Cognome FROM Giocatori WHERE Nome LIKE 'Francesc_'
```

restituisce rispettivamente le righe della tabella Giocatori, in cui il campo Nome inizia per Francesc ed è seguito da una singola lettera: quindi include sia Francesco sia Francesca, ma non Franceschino.

LABORATORIO

IL PROBLEMA Visualizza la targa, il modello e il prezzo per tutte le auto la cui targa presenta la lettera B in seconda posizione.

L'ANALISI Nel comando SELECT imposta la clausola LIKE sul campo Targa. Le targhe estratte potranno presentare qualsiasi carattere in prima posizione ('_, o ? in Access), la lettera B in seconda posizione seguita da una qualsiasi sequenza di caratteri ('%, o * in Access).

LA CODIFICA SQL

```
SELECT targa, modello, marca, prezzo  
FROM Auto  
WHERE targa Like '_B%';
```

in ambiente Access:

```
SELECT targa, modello, prezzo  
FROM Auto  
WHERE targa LIKE "?B*";
```

IL RISULTATO

targa	modello	prezzo
AB001LX	Xsara	14000
AB342CH	Panda	9000
CB401ET	Punto	15000
CB453ER	Micra	13000

BETWEEN

L'operatore BETWEEN controlla se un valore è compreso all'interno di un intervallo di valori, inclusi gli estremi. È possibile specificare, anteponendo lo a BETWEEN, anche l'operatore logico NOT per valutare la condizione opposta, cioè per controllare se il valore non rientra nell'intervallo specificato. Nota che il campo può essere di qualsiasi tipo: numerico, stringa, data... .

```
SELECT Nome_prodotto, Prezzo  
FROM Prodotti  
WHERE Prezzo BETWEEN 10 and 15
```

Nell'esempio illustrato vengono visualizzati i campi Nome_prodotto e Prezzo della tabella Prodotti, selezionando solo i prodotti che hanno il prezzo unitario compreso tra 10 e 15 euro.

LABORATORIO

IL PROBLEMA Visualizza la targa, il modello e il prezzo delle auto con prezzo compreso tra 10.000 e 13.000 euro.

L'ANALISI Puoi risolvere questo problema utilizzando l'operatore BETWEEN all'interno del comando SELECT.

LA CODIFICA SQL

```
Select targa, modello, prezzo  
From Auto  
WHERE prezzo BETWEEN 10000 and 13000
```

IL RISULTATO

targa	modello	prezzo
DH306HD	Punto	13000
CF786JY	Clio	11000
CB453ER	Micra	13000

Operatori logici

Come già accennato, le condizioni possono essere complesse se vengono congiunte condizioni semplici con gli operatori AND, OR e NOT. Nel seguente sono mostrati due esempi di condizioni complesse.

```
SELECT Nome_prodotto, Prezzo  
FROM Prodotti  
WHERE Nome_prodotto LIKE 'C%' AND Prezzo < 15
```

Visualizza i campi Nome_prodotto e Prezzo, appartenenti alla tabella Prodotti, selezionando solo i record che hanno il nome che incomincia con la lettera C e un prezzo inferiore ai 15 euro.

```
SELECT Nome_prodotto, Prezzo  
FROM Prodotti  
WHERE (Nome_prodotto LIKE 'C%' OR  
Nome_prodotto LIKE 'P%') AND Prezzo < 10
```

Visualizza i campi Nome_prodotto e Prezzo, appartenenti alla tabella Prodotti, selezionando solo i record che hanno il nome che inizia con la lettera C o con la lettera P (criterio OR) e prezzo inferiore a 10 euro (criterio AND).

IN

L'operatore IN controlla se un valore appartiene a un insieme specificato di valori.

Anche IN può essere preceduto da NOT per indicare la condizione opposta, cioè la non appartenenza del valore all'insieme dei valori.

```
SELECT *  
FROM Fornitori  
WHERE Città IN ('Londra', 'Roma', 'Parigi')
```

Il risultato di questa query visualizza tutti i campi della tabella Fornitori, selezionando solo i record residenti a Londra, Roma o Parigi.

DISTINCT

Per eliminare i valori ripetuti viene utilizzata la clausola DISTINCT. I record duplicati verranno mostrati una sola volta.

```
SELECT DISTINCT Ruolo  
FROM Giocatori
```

Il risultato dell'esempio visualizza tutti i ruoli dei giocatori presenti nella tabella Giocatori: le righe duplicate vengono eliminate.

LABORATORIO

IL PROBLEMA Visualizza tutte le cilindrate presenti nel database.

L'ANALISI Si tratta di visualizzare **una sola volta** il valore di un campo che si ripete all'interno della tabella. Per visualizzare le cilindrate una sola volta eliminando i duplicati occorre inserire nel comando SELECT la clausola DISTINCT.

LA CODIFICA SQL

```
SELECT DISTINCT (cilindrata)  
FROM Auto
```

IL RISULTATO

cilindrata
999
1250
1350
1600
2000

■ Ordinare i risultati di una query

La clausola ORDER BY ordina i dati visualizzati secondo uno o più campi specificati, in ordine crescente o decrescente. L'ordinamento è crescente per default; se lo si vuole decrescente, bisogna specificare DESC accanto agli attributi sui quali si vuole eseguire l'ordinamento.

Nell'esempio seguente il risultato della query sarà una tabella con i nomi e i relativi prezzi dei prodotti ordinati in modo crescente in base al nome del prodotto.

```
SELECT Nome_prodotto, Prezzo  
FROM Prodotti  
ORDER BY Nome_prodotto
```

Nella clausola ORDER BY possono essere specificati più campi: in questo caso l'ordinamento avviene sul primo campo e nel caso in cui diverse righe presentino lo stesso valore si ha anche l'ordinamento sul secondo campo.

Scrivendo, per esempio, "ORDER BY cognome, nome" si ha l'ordinamento in base al cognome e, a parità di cognome, si ha l'ordinamento sul campo nome.

LABORATORIO

IL PROBLEMA Visualizza la targa, il modello e il colore per tutte le auto di modello Xsara o Punto o Clio ordinate per targa.

L'ANALISI Nel comando SELECT imposta la condizione WHERE per estrarre le righe che presentano il valore di modello compreso nell'insieme 'Xsara', 'Punto', 'Clio'. Con l'aggiunta della clausola ORDER BY le targhe sono ordinate alfabeticamente.

LA CODIFICA SQL

```
SELECT targa, modello, colore  
FROM Auto  
WHERE modello IN ('Xsara', 'Punto', 'Clio')  
ORDER BY targa
```

La condizione:

```
modello IN ('Xsara', 'Punto', 'Clio')
```

è equivalente a:

```
modello = 'Xsara' OR modello = 'Punto' OR modello = 'Clio'
```

IL RISULTATO

targa	modello	colore
AB001LX	Xsara	grigio
CB401ET	Punto	blu
CF786JY	Clio	verde
DH306HD	Punto	rosso
EF786JY	Clio	nero

FISSA LE CONOSCENZE

1. Quando si usa l'istruzione SELECT?
2. Quali dati posso essere estratti con l'istruzione SELECT * FROM Auto?
3. A quali condizioni è equivalente la clausola NOT BETWEEN?
4. A quali condizioni è equivalente il predicato NOT IN?
5. A che cosa servono e quali sono i caratteri jolly?
6. Quando si usa la clausola DISTINCT?

5. ALTRI USI DELL'ISTRUZIONE SELECT

Nelle applicazioni può capitare di dover fare calcoli al volo sui campi dei record estratti: per esempio, se occorre visualizzare un prezzo convertito in vari cambi (euro, dollari, sterline...), oppure quando bisogna calcolare una percentuale o una frazione.

Vediamo due esempi.

LABORATORIO

IL PROBLEMA Visualizza la targa, il modello e il prezzo in euro e in dollari di tutte le auto Clio con prezzo maggiore di 1.200 euro.

L'ANALISI Per risolvere questo problema occorre inserire una **condizione doppia** nella clausola WHERE del comando SELECT, in modo da estrarre dalla tabella tutte le righe che soddisfano le condizioni richieste. Inoltre, per calcolare il prezzo in dollari devi moltiplicare il prezzo per 1.17 e assegnare il nome Dollari al campo calcolato utilizzando la clausola AS.

LA CODIFICA SQL

```
SELECT targa, modello, prezzo, prezzo * 1.17 AS dollari  
FROM Auto  
WHERE modello = 'Clio' and prezzo > 12000
```

IL RISULTATO

targa	modello	prezzo	dollari
EF786JY	Clio	14000	16380

LABORATORIO

IL PROBLEMA Visualizza la targa, il modello, il prezzo e il prezzo aumentato del 10% per tutte le auto la cui targa presenta la lettera B in seconda posizione, che sono di colore blu o rosso e che hanno prezzo compreso tra 10.000 e 20.000 euro.

L'ANALISI Nel comando SELECT, bisogna impostare la condizione WHERE per estrarre le righe che presentano targa LIKE '_B%', Colore = 'blu' o Colore = 'rosso' e Prezzo BETWEEN 10000 e 20000. Il nuovo prezzo è calcolato con la formula prezzo + (prezzo * 10/100)

LA CODIFICA SQL

```
SELECT targa, modello, prezzo, (prezzo + prezzo*10/100) AS nuovoPrezzo  
FROM Auto  
WHERE targa LIKE '_B%'  
AND colore IN ('blu','rosso')  
AND prezzo BETWEEN 10000 AND 20000
```

In ambiente Access la codifica è la seguente:

```
SELECT targa, modello, prezzo, (prezzo + prezzo*10/100) AS nuovoPrezzo  
FROM Auto  
WHERE targa LIKE '?B*'  
AND colore IN ('blu','rosso')  
AND prezzo BETWEEN 10000 AND 20000
```

La condizione:

```
prezzo BETWEEN 10000 AND 20000
```

è equivalente a:

```
prezzo >=10000 AND prezzo = 20000
```

IL RISULTATO

targa	modello	prezzo	nuovoPrezzo
CB401ET	Punto	15000	16500

■ Generare nuove tavole

Come abbiamo già detto, il risultato di una query è una tabella di tipo temporaneo, che scompare al termine dell'esecuzione. Se si desidera conservarla, è possibile **creare una tabella permanente** a partire dai risultati della query, usando la clausola INTO.

```
SELECT ListaCampi INTO NuovaTabella  
FROM TabellaOrigine  
{WHERE condizione}
```

Questa query genera la tabella NuovaTabella che contiene le colonne della tabella TabellaOrigine che compaiono in ListaCampi.

La tabella NuovaTabella potrà essere poi utilizzata all'interno delle query.

LABORATORIO

IL PROBLEMA Genera la tabella TabPunto contenente le targhe delle auto del modello Punto.

L'ANALISI Per risolvere questo problema, utilizza il comando SELECT con la clausola INTO.

LA CODIFICA SQL

```
SELECT targa, modello INTO TabPunto  
FROM Auto  
WHERE modello = 'Punto'
```

IL RISULTATO

targa	modello
CB401ET	Punto
DH306HD	Punto

6. L'OPERAZIONE JOIN

In un database le informazioni sono generalmente suddivise in più tabelle e, in molti casi, per ricostruire l'informazione complessiva occorre valutare insieme le tabelle. L'interrogazione deve quindi coinvolgere più tabelle. Per comprendere appieno come avviene un'interrogazione su più tabelle bisogna aver ben chiaro il concetto di **prodotto cartesiano**.

■ Il prodotto cartesiano

Il **prodotto cartesiano** è il modo più semplice per **combinare due** (o più) tabelle diverse. Esso corrisponde a una regola matematica che mette in relazione gli elementi di un insieme con quelli di un altro.

Il **prodotto cartesiano di due tabelle** restituisce una tabella con un numero di colonne pari alla somma del numero delle colonne delle tabelle. Ciascuna riga è ottenuta unendo ciascuna riga della prima tabella con ciascuna riga della seconda.

La tabella risultato di questa operazione ha un numero di righe pari al prodotto del numero di righe delle due tabelle.

Facciamo un esempio, ipotizzando di avere le seguenti tabelle.

Nazioni

nome	codContinente
Italia	1
Algeria	2
Francia	1

Continenti

idContinente	nome
1	Europa
2	Africa

Il loro prodotto cartesiano è il seguente:

nome	codContinente	idContinente	nome
Italia	1	1	Europa
Italia	1	2	Africa
Algeria	2	1	Europa
Algeria	2	2	Africa
Francia	1	1	Europa
Francia	1	2	Africa

Il risultato si ottiene prendendo la prima riga di Nazioni e affiancando tutte le righe di Continenti (si ottengono le prime due righe). Poi si ripete per la seconda riga di Nazioni e così via.

Otterremo quindi sei righe (poiché $3 \times 2 = 6$).

Il comando SQL che produce il prodotto cartesiano è:

```
SELECT *
FROM Nazioni, Continenti
```

■ JOIN

Non tutte le righe del prodotto cartesiano sono significative: nell'esempio infatti hanno significato solo quelle per cui il campo codContinent è uguale al campo idContinent, perché indicano il nome del continente in cui si trovano le nazioni:

nome	codContinent	idContinent	nome
Italia	1	1	Europa
Italia	1	2	Africa
Algeria	2	1	Europa
Algeria	2	2	Africa
Francia	1	1	Europa
Francia	1	2	Africa

Un modo per filtrare i dati del prodotto cartesiano così da ottenere solo le righe che hanno significato è quello di impostare un vincolo nella clausola WHERE:

```
SELECT *
FROM Nazioni, Continenti
WHERE Nazioni.codContinent = Continenti.idContinent
```

La giunzione interna (INNER JOIN) è quella che coincide con la query sopra descritta. Questa query (fondamentale) può essere espressa anche nel seguente modo:

```
SELECT *
FROM Nazioni INNER JOIN Continenti
ON Nazioni.codContinent = Continenti.idContinent
```

Le due query proposte sono equivalenti; la seconda però è più esplicita e più semplice da leggere. In caso di presenza di valori nulli in uno dei due campi non si mostrano le righe del prodotto cartesiano.

Per semplificare la scrittura e per rendere più leggibile il codice, è possibile far uso degli **Alias**. Nell'esempio si può indicare con N la tabella Nazioni e con C la tabella Continenti:

```
SELECT N.nome, C.nome
FROM Nazioni AS N, Continenti AS C
WHERE N.codContinent = C.idContinent
```

Riepilogando, le possibili sintassi per un'operazione di INNER JOIN sono:

```
SELECT *
FROM Tabella1, Tabella2
WHERE campoTabella1 = campoTabella2
```

e

```
SELECT *
FROM Tabella1 INNER JOIN Tabella2
ON campoTabella1 = campoTabella2
```

■ JOIN su più tabelle

Quando le informazioni da elaborare sono contenute su tre o più tabelle, si può estendere la JOIN a più tabelle:

```
SELECT *
FROM Tabella1, Tabella2, Tabella3
WHERE
    campoTabella1 = campoTabella2
AND
    campoTabella2 = campoTabella3
```

Rimane comunque sempre valida la seguente sintassi:

```
SELECT *
FROM Tabella1 INNER JOIN (Tabella2 INNER JOIN Tabella3
                           ON campoTabella2 = campoTabella3)
                           ON campoTabella1 = campoTabella2
```

Facciamo subito un esempio pratico, considerando anche la tabella Capitali.

Capitali

nazione	città
Italia	Roma
Algeria	Algeri
Francia	Parigi

Se vogliamo conoscere il nome del continente in cui sono situate le capitali, dobbiamo considerare tutte e tre le tabelle. La tabella Continenti è legata alla tabella Nazioni attraverso il codice del continente, mentre la tabella Nazioni è collegata con la tabella Capitali attraverso il nome della nazione:

```
SELECT Continenti.nome, Capitali.citta
FROM Continenti, Nazioni, Capitali
WHERE Continenti.idContinente = Nazioni.codContinente
AND Capitali.Nazione = Nazioni.Nome
```

LABORATORIO

IL PROBLEMA Visualizza la targa, il modello e la marca di tutte le auto.

L'ANALISI I dati da visualizzare sono contenuti in due tabelle diverse. Pertanto è necessario congiungere le due tabelle interessate (Auto, Modello), usando la chiave esterna Modello per collegare la tabella Auto con la tabella Modello che ha come chiave primaria nomeModello.

LA CODIFICA SQL

```
SELECT targa, modello, marca  
FROM Auto, Modello  
WHERE Auto.modello = Modello.nomeModello
```

oppure:

```
SELECT targa, modello, marca  
FROM Auto INNER JOIN Modello  
ON Auto.modello = Modello.nomeModello
```

IL RISULTATO

targa	modello	marca
AB001LX	Xsara	Citroën
AB342CH	Panda	Fiat
BA666LL	33	Alfa Romeo
CB401ET	Punto	Fiat
DH306HD	Punto	Fiat
EF786JY	Clio	Renault
CF786JY	Clio	Renault
CB453ER	Micra	Nissan

Se invece si volessero conoscere solo i dati delle auto di marca Fiat, basterebbe aggiungere una condizione alla query precedente:

```
SELECT targa, modello, marca  
FROM Auto, Modello  
WHERE Auto.modello = Modello.nomeModello  
AND Modello.marca = 'Fiat'
```

targa	modello	marca
AB342CH	Panda	Fiat
CB401ET	Punto	Fiat
DH306HD	Punto	Fiat

LABORATORIO

IL PROBLEMA Per ogni auto visualizza la targa, il modello, la marca e la nazione dove è stata prodotta.

L'ANALISI I dati da visualizzare sono contenuti in tre tabelle diverse. Pertanto è necessario congiungere le tre tabelle interessate (Auto, Modello, Marca) utilizzando la chiave esterna Modello per collegare le tabelle Auto e Modello e la chiave esterna Marca per collegare le tabelle Modello e Marca.

LA CODIFICA SQL

```
SELECT A.targa, A.cilindrata, A.modello, MR.NomeMarca, MR.sede  
FROM Auto AS A, Modello AS ML, Marca AS MR  
WHERE A.modello = ML.nomeModello  
AND ML.Marca = MR.NomeMarca
```

oppure:

```
SELECT A.targa, A.cilindrata, A.modello, MR.NomeMarca, MR.sede  
FROM (Auto AS A INNER JOIN Modello AS ML  
      ON A.modello = ML.nomeModello)  
INNER JOIN Marca AS MR ON ML.Marca = MR.NomeMarca
```

IL RISULTATO

targa	cilindrata	modello	NomeMarca	sede
AB001LX	1600	Xsara	Citroën	Parigi
AB342CH	999	Panda	Fiat	Torino
BA666LL	1350	33	Alfa Romeo	Milano
CB401ET	2000	Punto	Fiat	Torino
DH306HD	1350	Punto	Fiat	Torino
EF786JY	1600	Clio	Renault	Parigi
CF786JY	1350	Clio	Renault	Parigi
CB453ER	1250	Micra	Nissan	Tokio

FISSA LE CONOSCENZE

1. Che cos'è il prodotto cartesiano di due tabelle?
2. Quale funzione ha la clausola WHERE in una JOIN?
3. Come si estende una JOIN su più tabelle?
4. Che cosa sono gli alias e a che cosa servono?
5. Quando si presenta la necessità di effettuare un'operazione JOIN tra due o più tabelle?
6. Dal prodotto cartesiano di due tabelle come si ottiene una JOIN?

7. TIPI DI JOIN

Abbiamo visto nella Lezione precedente che quando dobbiamo selezionare dati da due o più tabelle, per avere dei risultati completi occorre effettuare delle operazioni JOIN.

Se con la INNER JOIN selezioniamo tutti i valori che accomunano le tabelle, nella OUTER JOIN avremo come risultato anche le righe che non trovano corrispondenza. A sua volta l'OUTER JOIN può essere di tre tipi:

- LEFT JOIN;
- RIGHT JOIN;
- SELF JOIN.

■ LEFT JOIN

LEFT JOIN visualizza tutte le righe della prima tabella (quella a sinistra dell'operatore). Inoltre congiunge le righe della prima tabella con quelle della seconda tabella, che hanno valori corrispondenti.

In altre parole, se c'è una corrispondenza tra le due tabelle vengono riportati i dati di entrambe le tabelle, se non c'è vengono comunque riportati i dati della tabella di sinistra.

```
SELECT *
FROM Tabella1 LEFT JOIN Tabella2
ON campoTabella1 = campoTabella2
```

Consideriamo le due tabelle Studenti e Voti.

La prima contiene i dati di alcuni studenti, la seconda i voti degli studenti in alcune prove:

- **Studenti** (matricola, nome, cognome);
- **Voto** (matricola, voto, materia).

Studenti

matricola	nome	cognome
m1	Mario	Rossi
m3	Sara	Bianchi
m4	Andrea	Bruni

Voti

matricola	voto	materia
m1	8	italiano
m1	6	matematica
m3	7	fisica
m3	8	italiano

La query seguente riporta tutti i record della tabella Studenti, anche se gli studenti non hanno avuto valutazioni:

```
SELECT *
FROM Studenti as S LEFT JOIN Voti as V
ON S.matricola = V.matricola
```

S.matricola	nome	cognome	V.matricola	voto	materia
m1	Mario	Rossi	m1	6	matematica
m1	Mario	Rossi	m1	8	italiano
m3	Sara	Bianchi	m3	8	italiano
m3	Sara	Bianchi	m3	7	fisica
m4	Andrea	Bruni	null	null	null

Se poi si vogliono ottenere solo le righe relative agli studenti che non hanno valutazioni, bisogna aggiungere alla query precedente la clausola che estrae le righe per le quali il campo V.matricola è vuoto (IS NULL):

```
SELECT *
FROM Studenti as S Left JOIN Voti as V
ON S.matricola = V.matricola
WHERE V.matricola IS NULL
```

S.matricola	nome	cognome	V.matricola	voto	materia
m4	Andrea	Bruni			

LABORATORIO

IL PROBLEMA Visualizza i nomi dei modelli per i quali non sono state vendute auto.

L'ANALISI Per risolvere questo problema bisogna individuare il nome dei modelli che sono presenti nella tabella Casa, ma non sono presenti nella tabella Auto: si tratta di effettuare un'operazione LEFT JOIN tra le tabelle Auto e Casa e poi selezionare solo le righe che ci interessano (il campo modello della tabella Auto deve essere nullo).

LA CODIFICA SQL

```
SELECT M.nomeModello
FROM Modello AS M LEFT JOIN Auto AS A
ON M.nomeModello = A.modello
WHERE A.modello IS NULL
```

IL RISULTATO

nomeModello
600
Giulietta

■ RIGHT JOIN

RIGHT JOIN visualizza tutte le righe della seconda tabella (quella a destra dell'operatore), congiungendo con le righe della prima solo le righe della seconda per le quali si trovano valori corrispondenti nella prima tabella.

In altre parole, se c'è una corrispondenza tra le due tabelle vengono riportati i dati di entrambe le tabelle, se non c'è vengono comunque riportati i dati della tabella di destra:

```
SELECT *
FROM Tabella1 RIGHT JOIN Tabella2
ON campoTabella1 = campoTabella2
```

Si possono ottenere i dati degli studenti senza valutazioni con il comando:

```
SELECT *
FROM Voti as V RIGHT JOIN Studenti as S
ON S.matricola = V.matricola
WHERE V.matricola IS NULL
```

V.matricola	voto	materia	S.matricola	nome	cognome
null	null	null	m4	Andrea	Bruni

■ SELF JOIN

SELF JOIN consente di unire una tabella a se stessa.

Si può utilizzare una SELF JOIN quando si desidera creare un insieme di risultati che unisca record in una tabella ad altri record nella stessa tabella. Per elencare una tabella due volte nella stessa query, è necessario specificare un alias di tabella, come nel seguente codice:

```
SELECT T1.campo1, T2.campo2
FROM tabella_uno AS T1, tabella_uno AS T2
WHERE T1.campo1 = T2.campo2
```

Facciamo un esempio. Supponiamo di avere la seguente tabella Genitori:

genitore	figlio
Luca	Anna
Maria	Anna
Giorgio	Luca
Silvia	Maria
Enzo	Maria

Se si vogliono conoscere i nonni di Anna, si può impostare la query seguente, che comporta impostare gli alias G1 e G2 della tabella Genitori.

```
SELECT G1.Genitore AS Nonni
FROM Genitori AS G1, Genitori AS G2
WHERE G1.figlio = G2.genitore
AND G2.figlio = 'Anna'
```

Il risultato sarà quello mostrato nella tabella.

Nonni
Giorgio
Enzo
Silvia

LABORATORIO

IL PROBLEMA Visualizza il nome e il cognome del capo del dipendente di cognome Verdi. La tabella Dipendenti contiene codice, cognome, nome e codice del capo dei dipendenti di una ditta.

Codice	Cognome	Nome	Codice_capo
1	Rossi	Mario	6
2	Verdi	Luigi	5
3	Bianchi	Maria	6
4	Bruni	Clara	5
5	Neri	Luca	4
6	Conti	Filippo	6

L'ANALISI Per risolvere questo problema bisogna utilizzare un'operazione SELF JOIN, perché il capo di Verdi è a sua volta un dipendente presente nella tabella Dipendenti. Viene quindi eseguita un'operazione SELF JOIN tra la tabella Dipendenti alias T1 e la tabella Dipendenti alias T2. Viene poi selezionata solo la riga cui si è interessati (`T1.Cognome = 'Verdi'`).

LA CODIFICA SQL

```
SELECT T2.Nome, T2.Cognome  
FROM Dipendenti AS T1, Dipendenti AS T2  
WHERE T1.Codice_capo = T2.Codice  
AND T1.Cognome = 'Verdi'
```

IL RISULTATO

Nome	Cognome
LUCA	NERI

FISSA LE CONOSCENZE

1. Quali sono i tipi dell'operazione JOIN?
2. Che differenza c'è tra INNER JOIN e OUTER JOIN?
3. Quali dati sono selezionati con la LEFT JOIN?
4. Quale clausola bisogna definire perché siano estratti solo i dati che compaiono nella tabella di destra, ma non in quella di sinistra?
5. Quando si usa la SELF JOIN?
6. Che cosa visualizza l'operazione RIGHT JOIN?
7. Quando è necessaria la clausola IS NULL nelle OUTER JOIN?

8. FUNZIONI DI AGGREGAZIONE

All'interno del comando SELECT possono essere utilizzate alcune funzioni predefinite, che agiscono sui valori contenuti in insiemi di righe della tabella (da cui il nome **funzioni di aggregazione**):

- COUNT() conta il numero di righe di una tabella;
- SUM() calcola la somma dei valori assunti da una colonna di una tabella;
- AVG() calcola la media dei valori assunti da una colonna di una tabella;
- MAX() calcola il massimo dei valori assunti da una colonna di una tabella;
- MIN() calcola il minimo dei valori assunti da una colonna di una tabella.

■ Funzione COUNT

La funzione COUNT **conta il numero di righe** presenti selezionate. La sua sintassi è la seguente:

```
SELECT COUNT (*)  
FROM NomeTabella  
[WHERE condizione]
```

oppure:

```
SELECT COUNT (nomeColonna)  
FROM NomeTabella  
[WHERE condizione]
```

La sintassi del linguaggio SQL richiede di specificare come argomento della funzione il carattere * (asterisco), oppure il nome di un attributo: nel primo caso la funzione calcola il numero delle righe della tabella, incluse quelle con campi di tipo NULL; nel secondo caso non vengono conteggiate le righe che hanno valore NULL nella colonna specificata tra le parentesi.

Il seguente comando restituisce il numero di tutte le righe presenti nella tabella Dipendenti:

```
SELECT COUNT (*)  
FROM Dipendenti
```

Specificando invece il nome dell'attributo (cellulare) come argomento della funzione COUNT, si ottiene il numero di persone per le quali è stato memorizzato il numero di cellulare:

```
SELECT COUNT (cellulare)  
FROM Dipendenti
```

Se si utilizza una clausola WHERE, la funzione COUNT restituisce il numero delle righe che soddisfano la condizione specificata. Con la query:

```
SELECT COUNT(*)  
FROM Dipendenti  
WHERE nome = 'Giovanni'
```

viene restituito il numero di dipendenti che si chiamano Giovanni. Il risultato del conteggio può essere anche descritto con un'opportuna intestazione aggiungendo la clausola AS:

```
SELECT COUNT(*) As 'Nome Giovanni'  
FROM Dipendenti  
WHERE nome = 'Giovanni'
```

Si può ottenere anche il numero di righe diverse usando DISTINCT. Il comando seguente restituisce quanti nomi diversi sono presenti nella tabella Dipendenti:

```
SELECT COUNT(DISTINCT nome)  
FROM Dipendenti
```

Attenzione: non tutti i DBMS supportano quest'ultima sintassi per la funzione COUNT.

■ Funzione SUM

La funzione SUM restituisce la **somma di tutti i valori contenuti in una colonna** (naturalmente di tipo numerico) specificata come argomento della funzione.

La sintassi nel linguaggio SQL è:

```
SELECT SUM(nomeColonna)  
FROM NomeTabella
```

La query seguente restituisce la somma degli stipendi di tutti i dipendenti:

```
SELECT SUM(stipendio)  
FROM Dipendenti
```

Nel caso siano presenti valori NULL, vengono considerati come 0.

Se si utilizza una clausola WHERE, la funzione prenderà in esame solo le righe che soddisfano la condizione specificata.

La seguente interrogazione restituisce la somma degli stipendi di tutti i dipendenti che lavorano nel dipartimento vendite:

```
SELECT SUM(stipendio)  
FROM Dipendenti  
WHERE dipartimento = 'Vendite'
```

■ Funzione AVG

La funzione AVG (dall'inglese average) calcola la **media aritmetica** dei valori numerici contenuti in una colonna. La sintassi nel linguaggio SQL è:

```
SELECT AVG(nomeColonna)  
FROM NomeTabella
```

La funzione non include nel calcolo i valori di tipo NULL presenti nella

colonna.

Nell'esempio viene calcolata la media degli stipendi del reparto vendite:

```
SELECT AVG(stipendio)
FROM Dipendenti
WHERE dipartimento = 'Vendite'
```

■ Funzione MIN

La funzione MIN restituisce il **valore minimo** tra i valori della colonna.

La sintassi nel linguaggio SQL è:

```
SELECT MIN(nomeColonna)
FROM nomeTabella;
```

Anche in questo caso, specificando la clausola WHERE, si restringe il calcolo alle righe che soddisfano la condizione.

La seguente query estrae lo stipendio minimo per i dipendenti che lavorano nel reparto vendite:

```
SELECT MIN(stipendio) AS 'Stipendio minimo'
FROM Dipendenti
WHERE dipartimento = 'Vendite'
```

L'argomento della funzione MIN può essere anche di tipo stringa:

```
SELECT MIN(cognome)
FROM Dipendenti
```

In questo caso è estratto il primo cognome in ordine alfabetico.

■ Funzione MAX

La funzione MAX cerca il più grande dei valori numerici contenuti in una determinata colonna di una tabella. La sintassi nel linguaggio SQL è:

```
SELECT MAX(nomeColonna)
FROM nomeTabella;
```

La seguente query estrae lo stipendio massimo per i dipendenti che lavorano nel reparto vendite:

```
SELECT MAX(stipendio) AS 'Stipendio massimo'
FROM Dipendenti
WHERE dipartimento = 'Vendite'
```

È anche possibile utilizzare più funzioni di aggregazione contemporaneamente.

Con la seguente query vengono estratti i valori massimo e minimo degli stipendi, nonché lo stipendio medio dei dipendenti della ditta:

```
SELECT MAX(stipendio), MIN(stipendio), AVG(stipendio)
FROM Dipendenti
```

LABORATORIO

IL PROBLEMA Stampa il numero totale delle auto.

L'ANALISI Per risolvere questo problema va utilizzata la funzione di aggregazione COUNT. Il risultato di questa query è una tabella che ha un'unica riga e un'unica colonna (Numero).

LA CODIFICA SQL

```
SELECT COUNT(*) AS Numero  
FROM Auto
```

IL RISULTATO

Numero
8

LABORATORIO

IL PROBLEMA Visualizza la media dei prezzi, il prezzo più alto e quello più basso.

L'ANALISI Per risolvere questa interrogazione vengono utilizzate le funzioni AVG(Prezzo), MAX(Prezzo) e MIN(Prezzo). Il risultato di questa query è una tabella che ha una riga e tre colonne: Media, Massimo e Minimo.

LA CODIFICA SQL

```
SELECT AVG(prezzo) AS Media, MAX(prezzo) AS Massimo, MIN(prezzo) AS Minimo  
FROM Auto
```

IL RISULTATO

Media	Massimo	Minimo
16750	45000	9000

LABORATORIO

IL PROBLEMA Visualizza la somma dei prezzi delle auto Fiat.

L'ANALISI Viene utilizzata la funzione di aggregazione SUM(prezzo) insieme alla clausola WHERE marca = 'Fiat'. Il risultato di questa query è una tabella che ha una sola riga e una sola colonna, che si chiama Somma.

LA CODIFICA SQL

```
SELECT SUM(prezzo) AS Somma  
FROM Auto, Modello  
WHERE Auto.modello = Modello.nomeModello  
AND Modello.Marca = 'Fiat'
```

IL RISULTATO

Somma
37000

9. RAGGRUPPAMENTI

La clausola GROUP BY raggruppa le righe in base ai valori uguali delle colonne specificate. Questa opzione produce una riga di risultato per ogni raggruppamento.

```
SELECT elencoCampi  
FROM NomeTabella  
[WHERE condizione]  
GROUP BY elencoCampi  
[ORDER BY elencoCampi]
```

Scrivendo quindi:

```
SELECT reparto  
FROM Dipendenti  
GROUP BY reparto
```

si ottiene una tabella che ha tante righe quanti sono i reparti contenuti nella tabella Dipendenti.

In effetti, nella sua forma di utilizzo più semplice, la clausola GROUP BY produce un risultato analogo a SELECT DISTINCT. Lo stesso risultato si sarebbe ottenuto scrivendo:

```
SELECT DISTINCT reparto  
FROM Dipendenti
```

■ GROUP BY e le funzioni di aggregazione

La clausola GROUP BY viene di frequente usata con le funzioni di aggregazione: insieme consentono di effettuare calcoli su campi numerici raggruppati, per estrarre il valore medio, massimo, minimo, la somma, ecc. Per conoscere il numero di dipendenti per ciascun reparto scriviamo:

```
SELECT codReparto, COUNT (*)  
FROM Dipendenti  
GROUP BY codReparto
```

Questa query fornisce per ogni reparto il codice e il numero di dipendenti che lavorano nel reparto. La tabella risultante ha tante righe quanti sono i reparti e due colonne (codice del reparto e numero dipendenti).

In alcuni ambienti, come per esempio Access e SQL Server, nella clausola GROUP BY devono comparire tutti i nomi dei campi che compaiono accanto alla parola SELECT. Scrivendo:

```
SELECT codReparto, nomeReparto COUNT (*)  
FROM Dipendenti  
GROUP BY codReparto
```

viene generato un errore, perché il campo nomeReparto non compare nella clausola GROUP BY.

All'interno della query possono comparire più funzioni di aggregazione. Per esempio, scrivendo:

```
SELECT codReparto, MAX(stipendio), MIN(stipendio),
       FROM Dipendenti
      GROUP BY codReparto
```

viene visualizzato lo stipendio massimo e minimo per ciascun reparto.

LABORATORIO

IL PROBLEMA Visualizza, per ogni modello di auto, la media dei prezzi.

L'ANALISI Viene utilizzata la funzione di aggregazione AVG(prezzo) che, combinata alla clausola GROUP BY modello, calcola la media dei prezzi delle auto per ogni modello. Il risultato di questa query è una tabella che ha tante righe quanti sono i modelli e due colonne: modello e Media.

LA CODIFICA SQL

```
SELECT modello, AVG(prezzo) AS Media
      FROM Auto
     GROUP BY modello
```

IL RISULTATO

modello	Media
33	45000
Clio	12500
Micra	13000
Panda	9000
Punto	14000
Xsara	14000
Renault	22000

LABORATORIO

IL PROBLEMA Per ogni marca di automobile visualizza il numero di esemplari venduti.

L'ANALISI Si usa la funzione COUNT che, in JOIN tra le tabelle Auto e Modello e combinata alla clausola GROUP BY marca, determina il numero di auto vendute per ogni modello.

LA CODIFICA SQL

```
SELECT Modello.marca, Count(*) As Numero
      FROM Auto, Modello
     WHERE Modello.nomeModello = Auto.modello
    GROUP BY Modello.marca
```

oppure:

```
SELECT Modello.marca, Count(*) As Numero
      FROM Auto INNER JOIN Modello
        ON Modello.nomeModello = Auto.modello
    GROUP BY Modello.marca
```

IL RISULTATO

marca	Numero
Alfa Romeo	1
Citroën	1
Fiat	3
Nissan	1
Renault	2

LABORATORIO

IL PROBLEMA Visualizza quanto è stato incassato per la vendita delle auto, raggruppando dati per nazione e ordinamento in ordine decrescente di importo.

L'ANALISI Viene utilizzata la funzione SUM all'interno di un'operazione JOIN tra le tabelle Auto, Modello e Marca. Con la clausola GROUP BY si ottengono poi i raggruppamenti per nazione, mentre, con la clausola ORDER BY SUM(Prezzo) DESC, i dati sono ordinati secondo valori decrescenti di incasso totale.

LA CODIFICA SQL

```
SELECT nazione, SUM(prezzo) As Incasso  
FROM Auto, Modello, Marca  
WHERE Modello.nomeModello = Auto.modello  
AND Modello.marca = Marca.nomeMarca  
GROUP BY nazione  
ORDER BY SUM(prezzo) DESC
```

oppure:

```
SELECT nazione, SUM(prezzo) As Incasso  
FROM (Auto INNER JOIN Modello  
      ON Auto.modello = Modello.nomeModello)  
INNER JOIN Marca R ON Modello.Marca = Marca.NomeMarca  
GROUP BY nazione  
ORDER BY SUM(prezzo) DESC
```

IL RISULTATO

nazione	Incasso
Italia	82000
Francia	39000
Giappone	13000

■ Clausola HAVING

La clausola HAVING è simile alla clausola WHERE nell'istruzione SELECT, ma, mentre WHERE opera sulla singola riga di una tabella, HAVING agisce su **gruppi di righe**, che prima sono stati selezionati con una clausola GROUP BY.

```
SELECT codReparto, COUNT (*)  
FROM Dipendenti  
GROUP BY codReparto  
HAVING COUNT (*) < 10
```

Questo esempio visualizza il codice dei reparti in cui lavorano meno di dieci dipendenti.

Per decidere se specificare le condizioni usando WHERE o HAVING, la regola è semplice:

- se devi impostare la condizione su un campo della tabella, allora usa la clausola WHERE;
- se devi impostare la condizione su un campo di raggruppamento, la condizione concerne gli insiemi di tuple; allora devi usare la clausola HAVING.

Questo esempio visualizza il codice dei reparti dove lavorano più di 10 dipendenti che hanno lo stipendio maggiore di 1.500 euro. In questo caso si è applicata la clausola WHERE sul campo della tabella stipendio e la clausola HAVING sul campo di aggregazione COUNT(*) .

```
SELECT CodReparto, COUNT (*)  
FROM Dipendenti  
WHERE stipendio > 1500  
GROUP BY reparto  
HAVING COUNT (*) >10
```

LABORATORIO

IL PROBLEMA Visualizza il numero di auto per i modelli che hanno venduto almeno due esemplari.

L'ANALISI Per risolvere questo problema si utilizza la funzione di aggregazione COUNT con la clausola GROUP BY modello per calcolare il numero delle auto per ogni modello. Poi, applicando al risultato il criterio HAVING, si estraggono solo le righe che soddisfano il criterio di selezione. Il risultato di questa query è una tabella che ha tante righe quante sono i modelli che hanno almeno due auto e due colonne: Modello e Numero.

LA CODIFICA SQL

```
SELECT modello, COUNT(*) AS Numero  
FROM Auto  
GROUP BY modello  
HAVING COUNT(*) > 1
```

L'ESECUZIONE

Modello	Numero
Clio	2
Punto	2

FISSA LE CONOSCENZE

1. A che cosa serve la clausola GROUP BY?
2. Che differenza c'è tra la clausola HAVING e la clausola WHERE?
3. Se in una SELECT sono presenti due campi, quanti campi dovranno comparire nella clausola GROUP BY?
4. È possibile usare la clausola GROUP BY all'interno di un'operazione JOIN?

10. QUERY COMPLESSE

■ Interrogazioni nidificate

Un'interrogazione **nidificata** o subquery è una query inclusa in un'altra, ovvero un'interrogazione all'interno di altre interrogazioni.

Una subquery restituisce dati (tabelle o dati singoli) necessari all'esecuzione della query a un livello più alto: viene sempre eseguita prima la query più interna e, una volta completata, quelle situate al livello superiore.

Il caso più semplice di subquery è quella che restituisce un singolo valore che sarà confrontato con il risultato fornito dalla query esterna.

```
SELECT Campo1  
FROM Tabella1  
WHERE Tabella1.Campo1 [operatore di confronto]  
      (SELECT CampoY  
       FROM Tabella2  
       WHERE [condizione])
```

Questa query estrae i valori di Campo1 dalla tabella Tabella1 per i quali è soddisfatta la condizione di confronto con il valore di CampoY della tabella Tabella2. Per esempio la query:

```
SELECT Nome_prodotto, Prezzo  
FROM Prodotti  
WHERE Prezzo <  
      (SELECT AVG(Prezzo)  
       FROM Prodotti)
```

visualizza i campi Nome_prodotto e Prezzo appartenenti alla tabella Prodotti, selezionando solo quelli con il prezzo inferiore al prezzo medio di tutti i prodotti. In questo caso, prima è eseguita la query interna che restituisce un singolo valore (il prezzo medio), poi viene eseguita la query esterna che estrae tutti i prodotti che hanno un prezzo inferiore.

LABORATORIO

IL PROBLEMA Visualizza targa, modello e prezzo delle auto il cui prezzo è inferiore al prezzo medio.

L'ANALISI Devi definire una query per estrarre tutte le auto il cui prezzo è minore del risultato di un'altra query, che calcola la media dei prezzi.

LA CODIFICA SQL

```
SELECT targa, modello, prezzo  
FROM Auto  
WHERE prezzo < (SELECT AVG(prezzo) FROM Auto)
```

IL RISULTATO

targa	modello	prezzo
AB001LX	Xsara	14000
AB342CH	Panda	9000
CB401ET	Punto	15000
DH306HD	Punto	13000
EF786JY	Clio	14000
CF786JY	Clio	11000
CB453ER	Micra	13000

LABORATORIO

IL PROBLEMA Visualizza i dati dell'auto che ha prezzo massimo.

L'ANALISI Questa query usa le query annidate: la query interna determina il prezzo massimo, quella esterna estrae i dati delle auto che hanno il prezzo uguale al prezzo massimo.

LA CODIFICA SQL

```
SELECT A.modello, prezzo  
FROM Auto AS A, Modello as M  
WHERE A.modello = M.nomeModello  
AND A.prezzo = (SELECT MAX(prezzo)  
                FROM Auto)
```

IL RISULTATO

modello	prezzo
33	45000



LABORATORIO

IL PROBLEMA Visualizzare per quale modello è stata venduta l'auto diesel più cara.

L'ANALISI Questa query usa query annidate, impostando la condizione alimentazione = 'diesel' sia nella query esterna, sia in quella interna.

Infatti, se eliminassimo la condizione dalla query interna, troveremmo il prezzo maggiore tra tutte le auto e non solo tra quelle alimentate a diesel, mentre se la togliessimo dalla query esterna potrebbe venire inclusa un'auto non diesel, con quel prezzo.

LA CODIFICA SQL

```
SELECT A.modello  
FROM Auto AS A, Modello as M  
WHERE A.modello = M.nomeModello  
AND A.prezzo = (SELECT MAX(prezzo)  
                 FROM Auto  
                 WHERE alimentazione = 'diesel')  
AND alimentazione = 'diesel'
```

oppure:

```
SELECT A.modello  
FROM Auto AS A, Modello as M  
ON A.modello = M.nomeModello  
WHERE A.prezzo = (SELECT MAX(prezzo)  
                  FROM Auto  
                  WHERE alimentazione = 'diesel')  
AND alimentazione = 'diesel'
```

IL RISULTATO

modello
Punto

FISSA LE CONOSCENZE

1. In quale ordine viene eseguita una query complessa che contiene una query di primo livello e una query annidata?
2. Che cosa può restituire la query interna?
3. È corretta la query: Select Modello, Max(Prezzo) from Auto?
4. Definisci la query per visualizzare il modello e il prezzo dell'auto più costosa.
5. Definisci la query per visualizzare il modello e il prezzo dell'auto meno costosa tra quelle di marca Fiat.
6. Definisci la query per visualizzare quante auto costano meno del costo medio.

11. SUBQUERY COMPLESSE

Nel caso delle subquery è necessario fare una distinzione: esse infatti possono restituire un valore singolo (scalare), una singola riga, una singola colonna, oppure una tabella. Gli esempi della Lezione precedente rientrano nel primo caso, perché la query interna ha sempre restituito **un singolo valore** (la media dei prezzi, il prezzo massimo). Quando invece la subquery restituisce **tanti valori**, per esempio una colonna, essi possono essere usati per fare confronti attraverso gli operatori IN, ANY e ALL.

Predicato IN

Il predicato IN serve a controllare se il valore di un attributo è compreso tra quelli restituiti dalla subquery nidificata.

Il seguente esempio produce l'elenco delle province che appartengono a regioni che hanno più di cinque province:

```
SELECT NomeProvincia, NomeRegione  
FROM Provincia  
WHERE NomeRegione IN (SELECT NomeRegione  
                        FROM Provincia  
                        GROUP BY NomeRegione  
                        HAVING Count(*) > 5)
```

È possibile utilizzare NOT IN per estrarre solo le righe della tabella principale per le quali nessuna riga della tabella ottenuta con la subquery contiene un valore uguale.

LABORATORIO

IL PROBLEMA Per i modelli per i quali sono state vendute almeno due auto produci l'elenco delle auto (targa, modello e prezzo).

L'ANALISI Questa query può essere svolta utilizzando le query annidate: nella subquery vengono selezionati i modelli per i quali sono state vendute almeno due auto, nella query esterna vengono poi estratte le auto appartenenti a quei modelli.

LA CODIFICA SQL

```
SELECT targa, modello, prezzo  
FROM Auto  
WHERE modello IN (SELECT modello  
                   FROM Auto  
                   GROUP BY modello  
                   HAVING Count(*) >= 2)
```

IL RISULTATO

targa	modello	prezzo
CB401ET	Punto	15000
DH306HD	Punto	13000
EF786JY	Clio	14000
CF786JY	Clio	11000

Predicato ANY

Il predicato ANY restituisce **vero** se almeno uno dei valori restituiti dalla subquery soddisfa la condizione. Il predicato ANY invece risulta **falso** se la subquery restituisce un insieme vuoto, oppure se il confronto è falso per ciascuno dei valori restituiti dalla subquery.

Il seguente esempio produce la lista degli operai che guadagnano più di almeno un dirigente.

```
SELECT Nome, Stipendio  
FROM Operai  
WHERE Stipendio > ANY (SELECT Distinct Stipendio  
                        FROM Dirigenti)
```

LABORATORIO

IL PROBLEMA Visualizza le informazioni delle auto del modello Clio con prezzo superiore a quello di almeno un'auto del modello Punto.

L'ANALISI Questa query può essere svolta utilizzando le query annidate: nella subquery vengono estratti i prezzi delle auto del modello Clio, nella query esterna vengono poi estratte le auto del modello Punto con prezzo superiore ad almeno uno dei prezzi selezionati nella subquery.

LA CODIFICA SQL

```
SELECT targa, prezzo, cilindrata  
FROM Auto  
WHERE modello = 'Clio'  
AND prezzo > ANY (SELECT prezzo  
                   FROM Auto  
                   WHERE modello = 'Punto');
```

Predicato ALL

Il predicato ALL restituisce **vero** se il confronto è vero per ciascuno dei valori restituito dalla subquery, è **falso** se il confronto è falso per almeno uno tra i valori dell'elenco restituito dalla subquery.

Il seguente esempio produce la lista degli operai che non vivono in province in cui vive un dirigente.

```
SELECT Nome, Provincia  
FROM Operai  
WHERE Provincia <> ALL (SELECT Distinct Provincia  
                           FROM Dirigenti)
```

Il predicato NOT IN è equivalente a <> ALL.

Consideriamo le query:

```
SELECT Targa  
FROM Auto  
WHERE Prezzo > ANY  
(SELECT Prezzo  
FROM Auto)
```

```
SELECT Targa  
FROM Auto  
WHERE Prezzo >= ALL  
(SELECT Prezzo  
FROM Auto)
```

definite sulla tabella Auto di [fig. 9](#). La tabella in [fig. 10](#) definisce il valore di verità dei predicati ANY e ALL.

Targa	Prezzo
AS111EE	50000
BK446RF	100000
AJ767DH	90000

[fig. 9](#) Tabella Auto

ANY	ALL
F	F
V	V
V	F

[fig. 10](#) Predicati ANY e ALL

- ANY specifica che il risultato del confronto è **vero se è vero per una qualunque riga** del risultato dell'interrogazione.
- ALL specifica che il risultato del confronto è **vero se è vero per tutte le righe** del risultato dell'interrogazione.

LABORATORIO

IL PROBLEMA Visualizza targa e prezzo delle Punto che hanno un prezzo superiore a quello di tutte le CLIO.

L'ANALISI Questa query può essere svolta utilizzando le query annidate: nella subquery vengono estratti i prezzi delle auto del modello Clio, nella query esterna vengono poi estratte le auto del modello Punto con prezzo superiore a tutti i prezzi delle auto del modello Clio selezionato nella subquery.

LA CODIFICA SQL

```
SELECT targa, prezzo, cilindrata  
FROM Auto  
WHERE modello = 'Punto'  
AND prezzo > ALL (SELECT prezzo  
                  FROM Auto  
                  WHERE modello = 'Clio');
```

IL RISULTATO

targa	prezzo	cilindrata
CB401ET	15000	2000

Predicato EXISTS

Il predicato EXISTS controlla se vengono restituite righe dall'esecuzione della subquery: la condizione di ricerca è vera se la SELECT nidificata produce una o più righe come risultato, è falsa se la subquery restituisce un insieme vuoto.

Il seguente esempio riporta i nomi di tutti i lavoratori che sono anche nella tabella proprietari. Il predicato EXISTS riporta true se l'insieme di righe risultanti dalla subselect ne contiene almeno una, se invece l'insieme è vuoto, EXISTS riporta false.

```
SELECT Lavoratore  
FROM Aziende WHERE EXISTS  
    (SELECT * FROM Proprietari  
     WHERE Proprietari.Nome = Aziende.Lavoratore)
```

Facendo uso di NOT EXISTS è possibile verificare se la subquery non restituisce alcuna tupla.

LABORATORIO

IL PROBLEMA Visualizza per quali modelli non sono state vendute auto.

L'ANALISI Questa query può essere svolta usando le query annidate collegate con EXISTS, che restituirà un valore booleano TRUE se e solo se la subquery interna seleziona almeno una riga.

LA CODIFICA SQL

```
SELECT M.nomeModello  
FROM Modello AS M  
WHERE NOT EXISTS (SELECT *  
                   FROM Auto AS A  
                   WHERE A.modelo = M.nomeModello)
```

IL RISULTATO

nomeModello
600
Giulietta

Per conoscere per quali modelli è stata venduta almeno un'auto, usiamo il predicato EXISTS:

```
SELECT M.nomeModello  
FROM Modello AS M  
WHERE EXISTS (SELECT *  
              FROM Auto AS A  
              WHERE A.modelo = M.nomeModello)
```

IL RISULTATO

nomeModello
Xsara
Panda
Punto
Micra
Clio
33

LABORATORIO

IL PROBLEMA Visualizza, per ogni modello, la percentuale di auto vendute.

L'ANALISI In questo caso la subquery non compare all'interno della clausola WHERE, ma al denominatore del calcolo percentuale. Nella subquery si determina il numero totale di auto vendute.

LA CODIFICA SQL

```
SELECT Modello, Count(*) /  
       (SELECT Count(*) FROM Auto)  
      *100 AS Percentuale  
  FROM Auto  
 GROUP BY Modello
```

IL RISULTATO

Modello	Percentuale
33	12,5
Clio	25
Micra	12,5
Panda	12,5
Punto	25
Xsara	12,5

LABORATORIO

IL PROBLEMA Per ogni auto, determina lo scostamento tra il suo prezzo e il prezzo medio delle auto dello stesso modello.

L'ANALISI Bisogna sottrarre dal costo di ogni auto il prezzo medio delle auto della stessa categoria. Quest'ultimo è calcolato nella subquery e bisogna impostare due alias sulla tabella Auto (A1 e A2) perché nella subquery si possa far riferimento allo stesso campo (A2.modello) della query esterna (A1.modelo).

LA CODIFICA SQL

```
SELECT A1.targa, A1.modelo, A1.prezzo,  
       A1.prezzo - (SELECT AVG(A2.prezzo)  
                     FROM Auto AS A2  
                    WHERE A1.modelo  
                      = A2.modelo) AS scostamento  
  FROM Auto AS A
```

IL RISULTATO

targa	modello	prezzo	scostamento
AB001LX	Xsara	14000	0
AB342CH	Panda	9000	0
BA666LL	33	45000	0
CB401ET	Punto	15000	1000
DH306HD	Punto	13000	-1000
EF786JY	Clio	14000	1500
CF786JY	Clio	11000	-1500
CB453ER	Micra	13000	0

12. UNIONE, INTERSEZIONE E DIFFERENZA

■ Unione (UNION)

Il comando SQL UNION combina insieme i risultati di due tabelle o interrogazioni. Affinché questo operatore possa essere utilizzato correttamente è necessario che:

- le tabelle siano interrogate sul medesimo numero di colonne;
- le colonne richieste abbiano lo stesso nome;
- le colonne richieste nelle due tabelle abbiano dati omogenei.

La sintassi è la seguente:

[Istruzione SQL 1]

UNION

[Istruzione SQL 2]

[Order by campo]

LABORATORIO

IL PROBLEMA Date la tabella VenditeNegozio contenente i dati delle vendite effettuate nei negozi e la tabella VenditeInternet contenente i dati delle vendite effettuate tramite Internet, visualizza tutte le date (ordinate in modo decrescente) in cui è stata realizzata una vendita.

Tabella VenditeNegozio

Negozi	Importo	Data
Torino	1500	05-gen
Milano	250	07-gen
Roma	300	08-gen
Venezia	700	08-gen

Tabella VenditeInternet

Data	Importo
07-gen	250
10-gen	535
11-gen	320
12-gen	750

L'ANALISI Per realizzare questa funzione bisogna prelevare i dati dalle due tabelle VenditeNegozio e VenditeInternet con un'operazione UNION.

LA CODIFICA SQL

```
SELECT Data FROM VenditeNegozio  
UNION  
SELECT Data FROM VenditeInternet  
ORDER BY Data DESC
```

IL RISULTATO

Data
12-gen
11-gen
10-gen
08-gen
07-gen
05-gen

■ Intersezione (INTERSECT)

Come con il comando UNION, anche mediante INTERSECT è possibile intervenire su due istruzioni SQL. La differenza consiste nel fatto che, mentre UNION agisce fondamentalmente come un operatore di tipo OR, in cui il valore viene selezionato, sempre che questo sia presente nella prima o nella seconda istruzione, il comando INTERSECT funziona come un operatore AND, in cui il valore viene selezionato solo se questo appare in entrambe le istruzioni.

La sintassi è la seguente:

```
[Istruzione SQL 1]
INTERSECT
[Istruzione SQL 2]
```

LABORATORIO

IL PROBLEMA Utilizzando le tabelle dell'esercizio precedente, trova tutte le date in cui sono state effettuate vendite sia in negozio sia via Internet.

L'ANALISI Per realizzare questa operazione si esegue un'operazione INTERSECT sul risultato delle interrogazioni che forniscono le date dei giorni in cui si sono verificate delle vendite.

LA CODIFICA SQL

```
SELECT Data FROM VenditeNegozio
INTERSECT
SELECT Data FROM VenditeInternet
```

IL RISULTATO

Data
07-gen



■ Differenza (MINUS)

Questo comando esegue la **differenza tra i risultati prodotti da due interrogazioni**. Mediante questo comando vengono presi tutti i risultati restituiti dalla prima query ai quali vengono sottratti tutti quelli che sono presenti nella seconda query. Se nella seconda istruzione SQL sono inclusi risultati che non sono presenti nella prima istruzione, tali risultati vengono ignorati.

La sintassi è la seguente:

```
[Istruzione SQL 1]
```

```
MINUS
```

```
[Istruzione SQL 2]
```

In alcuni DBMS può essere utilizzato EXCEPT anziché MINUS. Per un corretto utilizzo, si consiglia di consultare la documentazione relativa al proprio DBMS.

LABORATORIO

IL PROBLEMA Utilizzando le tabelle VenditeNegozio e VenditeInternet di cui sopra, trova tutte le date in cui sono state registrate vendite in negozio, ma non via Internet.

L'ANALISI Per realizzare questa funzione bisogna utilizzare l'operatore MINUS tra la tabella VenditeNegozio e la tabella VenditeInternet, escludendo dalle righe estratte dalla prima interrogazione quelle estratte dalla seconda.

LA CODIFICA SQL

```
SELECT Data FROM VenditeNegozio  
MINUS  
SELECT Data FROM VenditeInternet
```

IL RISULTATO

Data
05-gen
08-gen

FISSA LE CONOSCENZE

1. Che differenza c'è tra il comando UNION e il comando INTERSECT?
2. In quali circostanze è possibile applicare il comando UNION?
3. Quando è possibile applicare il comando INTERSECT?
4. Quale risultato fornisce il comando MINUS?

13. LE VISTE

SQL riconosce due tipi di tabelle:

- le **tabelle base**, che sono le uniche in cui è possibile registrare i dati;
- le **tabelle derivate**, dette anche **viste (VIEW)**, che sono formule per combinare in una tabella virtuale i dati contenuti nella tabella base.

Grazie alle viste un utente ha una visione solo parziale dei dati contenuti nella tabella, e non può agire su dati che non lo riguardano.

La creazione di una vista avviene con la seguente istruzione:

```
CREATE VIEW nome-vista AS  
    istruzione-select
```

Il **contenuto di una vista** non viene memorizzato, ma **derivato** nel momento in cui la vista compare in un'istruzione. Esso, quindi, **coincide sempre con il contenuto delle tabelle base**: qualsiasi modifica ai dati base si riflette immediatamente sulla vista che accede ai dati.

Per cancellare una vista si ricorre all'istruzione:

```
DROP VIEW nome-vista
```

LABORATORIO

IL PROBLEMA Crea una vista con le auto prodotte a Torino.

L'ANALISI Bisogna creare una vista basata su una selezione e una congiunzione (tutte le auto le cui case costruttrici hanno sede a Torino).

LA CODIFICA SQL

```
CREATE VIEW AutoTorino AS  
    SELECT targa, prezzo, modello  
    FROM Auto, Modello, Marca  
    WHERE Auto.modello = Modello.nomeModello  
    AND Modello.Marca = Marca.NomeMarca  
    AND Marca.sede = 'Torino'
```

IL RISULTATO

targa	prezzo	modello
AB342CH	9000	Panda
CB401ET	15000	Punto
DH306HD	13000	Punto

LABORATORIO

IL PROBLEMA Usando la vista AutoTorino creata nell'esercizio precedente calcola il prezzo medio.

LA CODIFICA SQL

```
SELECT AVG(prezzo) AS Prezzo_Medio  
from AutoTorino
```

IL RISULTATO

Prezzo_Medio
12333,3333333333

LABORATORIO

IL PROBLEMA Visualizza per quale modello sono state vendute più automobili.

L'ANALISI Quando le query sono piuttosto complesse, la creazione di una vista può semplificare il lavoro, come nel caso proposto. Infatti per risolvere la query si può procedere creando prima una vista che contenga per ogni modello il numero di automobili vendute e poi cercare su questa vista il modello che ne ha vendute di più.

LA CODIFICA SQL

```
CREATE VIEW AutoTorino AS  
SELECT Modello.marca, count (*) as numero  
FROM Auto, Modello  
WHERE Auto.modello = Modello.nomeModello  
GROUP BY Modello.marca
```

IL RISULTATO

marca	numero
Alfa Romeo	1
Citroën	1
Fiat	3
Nissan	1
Renault	2

e successivamente:

LA CODIFICA SQL

```
SELECT nomeMarca  
FROM VistaMarca  
WHERE numero = (SELECT MAX(numero) FROM VistaMarca)
```

IL RISULTATO

marca
Fiat

FISSA LE CONOSCENZE

1. Che cosa sono le viste?
2. Con quale comando SQL si crea una vista?
3. Con quale comando SQL si elimina una vista?
4. Che differenza c'è tra una vista e una tabella?
5. Per quale motivo alle volte usare una vista può semplificare il lavoro di interrogazione di una base di dati?
6. Per quale motivo una vista mostra sicuramente i dati aggiornati?
7. I dati di una vista occupano spazio?

14. SICUREZZA DEI DATI

I comandi di tipo DCL (*Data Control Language*) controllano il livello di accesso che gli utenti hanno sugli oggetti del database.

Tali comandi sono:

- GRANT;
- REVOKE.

■ GRANT

Il comando GRANT permette all'amministratore della base di dati di fornire accessi diversificati alle tabelle in funzione del **tipo di utente** che ne fa richiesta.

La sintassi di questa istruzione utilizzata per assegnare determinati privilegi è la seguente:

```
GRANT [SELECT  
       INSERT  
       DELETE  
       UPDATE {nome colonna}  
       INDEX  
       ALTER  
       ALL]  
ON identificatore-tabella  
TO [nome-utente].... | PUBLIC
```

Il privilegio può essere:

- accordato a un solo utente;
- accordato a più utenti;
- concesso come PUBLIC. In quest'ultimo caso il privilegio è concesso a tutti gli utenti SQL.

■ REVOKE

L'istruzione REVOKE è concepita per rimuovere i privilegi di un utente assegnati con il comando GRANT.

La sua sintassi, prevedibilmente, è la seguente:

```
REVOKE [SELECT  
       INSERT  
       DELETE  
       UPDATE {nome colonna}  
       INDEX  
       ALTER  
       ALL]  
ON identificatore-tabella  
FROM [nome-utente].... | PUBLIC
```

LABORATORIO

IL PROBLEMA Definisci i privilegi di accesso in lettura e scrittura per l'utente Rossi alla tabella Auto.

L'ANALISI Vanno definiti tramite il comando GRANT i privilegi per l'utente "Rossi" (INSERT, SELECT, UPDATE) sulla tabella Auto.

LA CODIFICA SQL

```
GRANT INSERT, SELECT, UPDATE  
ON Auto  
TO Rossi;
```

LABORATORIO

IL PROBLEMA Togli all'utente "Rossi" i privilegi di scrittura sulla tabella Auto.

L'ANALISI Bisogna eliminare tramite il comando REVOKE i privilegi per l'utente "Rossi" (INSERT, UPDATE) sulla tabella Auto.

LA CODIFICA SQL

```
REVOKE INSERT, UPDATE  
ON Auto  
FROM Rossi;
```

I TRIGGER

Un TRIGGER (**regola attiva**) è un tipo speciale di procedura che viene eseguita automaticamente quando si verifica un evento nel database.

Spesso, soprattutto nei sistemi di grandi dimensioni, la modifica di determinati dati richiede un'azione automatica su altri dati registrati sullo stesso database o su database differenti e i TRIGGER (*trigger* in inglese vuol dire "grilletto") sono gli strumenti adatti a questo scopo. I TRIGGER vengono eseguiti, per esempio, quando un utente tenta di modificare dati tramite un comando di aggiornamento (istruzioni INSERT, UPDATE o DELETE) eseguito su una tabella o una vista.

Questi TRIGGER vengono attivati quando viene generato un evento valido, indipendentemente dal fatto che esistano o meno righe di tabella interessate.

I TRIGGER possono essere utilizzati in molte situazioni per:

1. mantenere l'integrità dei dati, controllando la validità dei valori inseriti;
2. segnalare automaticamente ad altri programmi che devono essere effettuate azioni quando vengono apportate modifiche a una tabella o vengono inseriti nuovi record (per esempio: aggiungere il contatto alla *mailing list* quando si registra un nuovo utente, inviare una email di conferma all'utente appena registrato);

3. creare tabelle di *auditing* per registrare i record che vengono modificati o eliminati, in modo da tenere traccia di chi ha eseguito la modifica e in quale data. Per esempio, può essere utile mantenere la storia delle modifiche apportate allo stipendio dei dipendenti di una ditta: tutte le volte che viene modificato uno stipendio viene registrata in una tabella del database una riga contenente il codice del dipendente, il vecchio stipendio, il nuovo stipendio, la data e l'ora dell'operazione.

Da un punto di vista generale, in un TRIGGER vengono specificati i tre elementi: EVENTO – CONDIZIONE – AZIONE.

A seguito della modifica specificata in EVENTO, se è vera la condizione CONDIZIONE, vengono specificate le azioni di AZIONE. La sintassi per creare un trigger in ambiente MySQL è la seguente:

1. Specifica il nome del TRIGGER.
2. Specifica se il TRIGGER deve essere attivato prima o dopo un'operazione.
3. Specifica l'evento che attiva il TRIGGER.
4. Specifica se il TRIGGER è set-oriented (attivato un'unica volta) oppure tuple-oriented (attivato tante volte quante sono le tuple interessate dall'operazione).
5. Specifica l'eventuale condizione che deve essere verificata per attivare il TRIGGER.
6. Specifica le azioni da eseguire.

- ```
1. CREATE TRIGGER <Nometrigger>
2. <BEFORE|AFTER>
3. <SELECT|DELETE|INSERT|UPDATE[(<Colonne>)]>
 ON <Nometabella>
4. [FOR EACH STATEMENT|FOR EACH ROW]
5. [WHEN <condition>]
6. <action>
```

Per un TRIGGER in update si può accedere al vecchio valore di attributo utilizzando la sintassi OLD.<colonna> e si può accedere al nuovo attributo utilizzando NEW.<colonna>.

La sintassi di CREATE TRIGGER può variare a seconda del DBMS utilizzato. Per eliminare un TRIGGER si usa il comando:

```
DROP TRIGGER trigger_name;
```

Per creare un TRIGGER in ambiente SQL Server la sintassi è la seguente:

1. Specifica il nome del TRIGGER.
2. Specifica il nome della tabella.
3. Opzioni che specificano quando il TRIGGER deve scattare: FOR indica un'azione contemporanea all'aggiornamento dei dati; AFTER indica un'azione successiva; INSTEAD OF indica che deve essere eseguita l'istruzione del TRIGGER al posto dell'istruzione SQL predefinita specificata dopo (INSERT, DELETE o UPDATE).
4. Specifica le azioni da eseguire.

- ```
1. CREATE TRIGGER nomeTrigger
2. ON nome Tabella
3. FOR / AFTER/ INSTEAD OF (INSERT, DELETE o UPDATE)
4. AS azione
```

Nel caso di una operazione di DELETE la tabella *Deleted* contiene le righe che sono state appena eliminate, mentre con una INSERT la tabella *Inserted* contiene le righe appena inserite. Nel caso di una operazione di UPDATE entrambe le tabelle contengono valori, perché la *Deleted* contiene i dati prima della modifica (le vecchie righe), mentre la *Inserted* contiene i dati dopo la modifica (le nuove righe).

LABORATORIO

IL PROBLEMA Crea un TRIGGER che permetta di effettuare il controllo sulla cilindrata quando una nuova auto è inserita nel database, ponendo Cilindrata = 900 se il valore inserito è minore di 0 e Cilindrata = 4800 se, invece, è stato inserito un valore maggiore di 4.800.

L'ANALISI In questo esempio il TRIGGER deve essere eseguito **prima** di inserire la riga nella tabella Auto e, se la cilindrata appena inserita (NEW.Cilindrata) è minore di 900, viene assegnato 900. Lo stesso procedimento viene adottato per mantenere il valore della cilindrata minore di 4.800.

LA CODIFICA MYSQL

```
CREATE TRIGGER NuovaAuto
BEFORE INSERT ON Auto
FOR EACH ROW
BEGIN
    IF NEW.Cilindrata < 0 THEN
        SET NEW.Cilindrata = 900
    END IF
    IF NEW.Cilindrata > 4800 THEN
        SET NEW.Cilindrata = 4800
    END IF
END
```

LA CODIFICA SQL SERVER

```
CREATE TRIGGER NuovaAuto
ON Auto
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @targa as VARCHAR(20)
    DECLARE @cilindrata as VARCHAR(20)
    DECLARE @prezzo as float
    DECLARE @modello as VARCHAR(20)
    DECLARE @colore as VARCHAR(20)
    DECLARE @marca as VARCHAR(20)
    DECLARE @alimentazione as VARCHAR(20)
    DECLARE @codProprietario as smallint

    SET @cilindrata = (select cilindrata from inserted)
    SET @targa = (select targa from inserted)
    SET @colore = (select colore from inserted)
    SET @prezzo = (select prezzo from inserted)
    SET @modello = (select modello from inserted)
    SET @marca = (select marca from inserted)
    SET @alimentazione = (select alimentazione from inserted)
    SET @codProprietario = (select codProprietario from inserted)
```

```

if(@cilindrata < 0)
    set @cilindrata = 900

if(@cilindrata > 4800)
    set @cilindrata = 4800

INSERT INTO Auto
    Values ( @targa , @cilindrata, @prezzo, @modello, @colore, @marca,
            @alimentazione,@codProprietario )
END;

```

In questo esempio, essendo specificata l'opzione INSTEAD OF INSERT, le istruzioni del TRIGGER vengono eseguite al posto dell'istruzione di INSERT di SQL; pertanto anche nel TRIGGER deve comparire l'istruzione INSERT.

LABORATORIO

IL PROBLEMA Aggiorna la tabella di log, inserendo nella tabella Storia il codice dell'auto, il vecchio prezzo e il nuovo prezzo tutte le volte che si aggiorna il prezzo di un'auto.

L'ANALISI In questo caso il TRIGGER deve essere eseguito dopo che è stata modificata una riga nella tabella Auto; in tal caso, se è stato modificato il prezzo e quindi il nuovo prezzo è diverso da quello vecchio, viene inserita una nuova riga nella tabella Storia contenente il codice, il prezzo vecchio e il prezzo nuovo.

LA CODIFICA MYSQL

```

CREATE TRIGGER AggiornaPrezzo
AFTER UPDATE ON Auto
FOR EACH ROW
BEGIN
    IF NEW.Costo <> OLD.Costo THEN
        INSERT INTO Storia
            Values (OLD.Codice, OLD.Costo, NEW.Costo)
    END IF
END

```

LA CODIFICA SQL SERVER

```

Create TRIGGER [dbo].[AggiornaPrezzo]
ON [dbo].[Auto] AFTER UPDATE
AS
    INSERT INTO Storico (targa, oldColore,newColore)
    (SELECT i.codice, d.costo, i.pre costo xxo
     FROM Inserted i
     INNER JOIN Deleted d ON i.targa = d.targa
     where i.costo <> d.costo)

```

15. LE TRANSAZIONI

Per **transazione** si intende una serie di istruzioni SQL che vengono tratte come se fossero una singola unità.

Esempio di transazione è il trasferimento di una somma da un conto corrente a un altro, che può essere implementato con i seguenti comandi SQL:

```
UPDATE CC  
SET Saldo = Saldo - 50  
WHERE Conto = 123
```

```
UPDATE CC  
SET Saldo = Saldo + 50  
WHERE Conto = 235
```

Per realizzare questa operazione sono necessari **due aggiornamenti separati**; i funzionari di banca vorranno essere sicuri o che entrambi questi aggiornamenti si verifichino, o che non si verifichi nessuno di essi: non vorrebbero certamente che, per un guasto di sistema, risultasse nel conto 235 un accredito di 50 euro e nessun addebito sul conto 123. C'è bisogno di una **garanzia** che se qualcosa va male nell'operazione, nessuno dei passi eseguiti abbia effetto.

Raggruppando gli aggiornamenti in una transazione abbiamo questa garanzia, perché una transazione può:

- andare completamente a buon fine;
- oppure
- fallire completamente, lasciando la base di dati intatta.

Le istruzioni di una transazione devono essere racchiuse tra i comandi **Begin Transaction** e **End Transaction**:

```
Begin Transaction  
    UPDATE CC  
        SET Saldo = Saldo - 50  
        WHERE Conto = 123  
    UPDATE CC  
        SET Saldo = Saldo + 50  
        WHERE Conto = 235  
End Transaction
```

Una transazione può avere solo due esiti:

1. **terminare correttamente**: questo avviene solo quando l'applicazione, dopo aver eseguito tutte le operazioni, esegue una particolare istruzione SQL, detta **COMMIT**, che garantisce la conclusione delle operazioni;
2. **terminare non correttamente** (anticipatamente) e quindi sono possibili 2 casi:
 - la **transazione** determina che è avvenuto un errore e interrompe le operazioni, eseguendo l'istruzione SQL **ROLLBACK**;
 - il **sistema** non è in grado di garantire la corretta prosecuzione della transazione (per esempio, per un guasto o a causa della violazione di un vincolo); la transazione viene quindi interrotta.

Se per qualche motivo la transazione non può terminare correttamente, il DBMS deve annullare (UNDO) le eventuali modifiche da essa apportate alla base di dati. Per esempio, può capitare che, dopo aver detratto 50 euro dal conto di codice 123, per un errore o in seguito a una cancellazione, il conto di codice 235 non sia presente: in questo caso si verificherebbe una situazione di incongruenza e, quindi, la modifica sul primo conto corrente dovrebbe essere annullata con un'operazione di ROLLBACK.

Il modello di transazioni usato dai DBMS è in realtà più articolato; in particolare è possibile definire dei cosiddetti SAVEPOINT, che vengono utilizzati da una transazione per annullare solo parzialmente il lavoro svolto.

Un SAVEPOINT serve a specificare un preciso punto all'interno di una transazione, giunti al quale sarà poi possibile effettuare un ROLLBACK.

Per definire un SAVEPOINT in MySQL si usa la sintassi:

```
SAVEPOINT <nome savepoint> ON ROLLBACK RETAIN CURSORS
```

e per eseguire un rollback parziale:

```
ROLLBACK TO SAVEPOINT <nome savepoint>
```

In SQL Server i rispettivi comandi sono:

```
SAVE TRANSACTION NomeSavePoint
```

e

```
ROLLBACK TRANSACTION NomeSavePoint
```

I comandi che agiscono sulle transazioni (COMMIT, ROLLBACK E SAVEPOINT) appartengono al TCL (*Transaction Control Language*).

Per chiarire meglio il funzionamento di questi comandi, utilizziamo come esempio la tabella Anagrafica, che presenta la seguente situazione:

Cognome	Stipendio
Rossi	20000
Bianchi	20000

Supponiamo ora di avere le seguenti istruzioni in ambiente MySQL:

```
INSERT INTO Anagrafica  
VALUES ('Neri', 35000);  
SAVEPOINT sp1;
```

```
INSERT INTO Anagrafica  
VALUES ('Viola', 35000);  
SAVEPOINT sp2;
```

```
INSERT INTO Anagrafica  
VALUES ('Marrone', 25000);
```

Le stesse istruzioni in ambiente SQL Server si scrivono:

```
INSERT INTO Anagrafica  
VALUES ('Neri',35000)  
SAVE TRANSACTION SavePointsp1
```

```
INSERT INTO Anagrafica  
VALUES ('Viola',35000)  
SAVE TRANSACTION SavePointsp2
```

```
INSERT INTO Anagrafica  
VALUES ('Marrone',25000)
```

A questo punto, la nostra tabella si presenterà così:

Cognome	Stipendio
Rossi	20000
Bianchi	20000
Neri	35000
Viola	35000
Marrone	25000

Se ora noi scrivessimo

```
ROLLBACK TO sp1; (MySQL)
```

```
Rollback Transaction SavePointsp1 (Sql Server)
```

avremmo:

Cognome	Stipendio
Rossi	20000
Bianchi	20000
Neri	35000

Se invece scrivessimo:

```
ROLLBACK TO sp2; (MySQL)
```

```
Rollback Transaction SavePointsp2 (Sql Server)
```

avremmo:

Cognome	Stipendio
Rossi	20000
Bianchi	20000
Neri	35000
Viola	35000

Se invece scrivessimo:

ROLLBACK;

avremmo come esito che l'intera transazione sarebbe annullata, quindi la nostra tabella si presenterebbe così:

Cognome	Stipendio
Rossi	20000
Bianchi	20000

Infine, scrivendo:

COMMIT;

l'intera transazione sarebbe consolidata (*committata*), tutti i SAVEPOINT sarebbero rimossi e, quindi, la nostra tabella si presenterebbe così:

Cognome	Stipendio
Rossi	20000
Bianchi	20000
Neri	35000
Viola	35000
Marrone	25000



FISSA LE CONOSCENZE

1. Che cos'è una transazione?
2. Quali esiti può avere una transazione?
3. Che cosa provoca il comando COMMIT?
4. Che cosa provoca il comando ROLLBACK?
5. Che cosa specifica un SAVEPOINT?



Creazione di database e tabelle

L'SQL è un **linguaggio non procedurale** utilizzato per creare, visualizzare, modificare tabelle di un **database relazionale**.

Attraverso l'istruzione CREATE TABLE è possibile creare una tabella in cui registrare dati, che possono essere **numerici** (interi e reali), **alfanumerici** (stringhe e date) oppure **oggetti**.

In una tabella è possibile **definire un indice** su un dato attributo, allo scopo di rendere più veloce il reperimento dei dati.

È inoltre possibile definire più indici sulla stessa tabella.

Per farlo si usa l'istruzione CREATE INDEX, che agisce su tabelle esistenti creando l'indice.

Modificare lo schema di una base di dati

Attraverso l'istruzione ALTER TABLE è possibile **modificare la struttura di una tabella** definita in precedenza aggiungendo campi, modificandone le caratteristiche, o cancellandoli.

Per **eliminare** una tabella (oppure un indice creato in precedenza) si usa l'istruzione DROP. Nel caso sia applicato a una tabella, il comando elimina sia la struttura sia tutte le righe in essa contenute.

Modificare i dati

È possibile **modificare il contenuto** dei campi di una tabella, oppure anche inserire o cancellare una riga della tabella stessa.

Per **inserire nuove righe** in una tabella si usa il comando INSERT.

Per **modificare i dati** precedentemente inseriti in un record si usa il comando UPDATE.

Per **cancellare i dati** di un record inserito si usa il comando DELETE.

L'istruzione SELECT

L'istruzione SELECT del linguaggio SQL consente di **interrogare la base di dati**.

La sua sintassi è la seguente:

```
SELECT identificatore-colonna  
FROM identificatore-tabella  
{WHERE condizione}
```

Una SELECT, quindi, preleva i dati di una colonna da una tabella se una condizione viene soddisfatta.

Il **risultato** di una SELECT è una **tabella** che successivamente può essere ancora elaborata usando la clausola ORDER BY.

Altri usi dell'istruzione SELECT

L'istruzione SELECT può essere usata per **implementare query complesse** e per visualizzare dati ottenuti da semplici calcoli effettuati sui campi della tabella.

JOIN

L'operatore JOIN serve a **unire le righe di due o più tabelle** quando i dati richiesti sono contenuti in tabelle diverse.

Per poter utilizzare questo operatore le tabelle interessate **devono avere un attributo in comune**.

Tipi di JOIN

Quando si devono estrarre dati da due tabelle, ma si vogliono considerare quelli che compaiono nella **prima tabella**, ma non nella **seconda**, si usano le operazioni di LEFT JOIN.

Quando si devono estrarre dati da due tabelle, ma si vogliono considerare quelli che compaiono nella **seconda tabella**, ma non nella **prima**, si usano le operazioni di RIGHT JOIN.

Con una SELF JOIN si uniscono le righe di una tabella ad altre righe della stessa tabella.

Funzioni di aggregazione

Insieme a SELECT possono essere usate alcune funzioni che agiscono sui valori contenuti in insiemi di righe.

- COUNT serve per **contare le righe**.
- MIN serve per determinare il **valore minimo** di una colonna.
- MAX serve per determinare il **valore massimo** di una colonna.
- SUM serve per calcolare la **somma dei valori** presenti in una colonna.
- AVG serve per determinare la **media dei valori** presenti in una colonna.

Raggruppamenti

La clausola GROUP BY del comando SELECT serve per **raggruppare i dati** in base a un dato campo.

Con la clausola HAVING è possibile selezionare i raggruppamenti che **soddisfano una determinata condizione**.

Query complesse

In SQL è possibile definire **interrogazioni annidate** tramite le subquery.

Una subquery è una **query inclusa in un'altra**, ovvero un'interrogazione all'interno di altre interrogazioni.

Una subquery restituisce un set di dati (tabelle o dati singoli) necessari all'esecuzione di query a un livello più alto.

Subquery complesse

Gli operatori IN, ANY, ALL sono necessari per **fare confronti** quando la query annidata restituisce un insieme di valori.

- IN serve a controllare se il valore di un campo è **compreso** tra quelli restituiti dalla subquery.
- ANY specifica che il confronto è vero, se è vero per **almeno una** riga della subquery.
- ALL specifica che il confronto è vero, se è vero per **tutte** le righe della subquery.

Unione, intersezione e differenza

Quando devono essere **uniti i risultati di due o più interrogazioni** si usano i seguenti operatori:

- UNION, che restituisce tutte le righe presenti **nella prima o nella seconda** interrogazione;
- INTERSECT, che restituisce tutte le righe presenti **sia nella prima sia nella seconda** interrogazione;
- MINUS, che restituisce tutte le righe presenti **nella prima ma non nella seconda** interrogazione.

Viste

Le viste sono **tabelle virtuali**: il contenuto di una vista non è memorizzato fisicamente nella base di dati, ma di essa è memorizzato il **comando che serve a crearla**.

Con le viste è possibile selezionare alcuni campi o alcune righe di una specifica tabella, in modo che l'utente possa **agire solo sui dati che gli interessano**.

Sicurezza dei dati

Il comando GRANT serve per **concedere permessi** diversi a utenti diversi (garantendo così la **sicurezza** dei dati). I TRIGGER servono per mantenere **l'integrità dei dati** del database: un TRIGGER è una procedura che viene eseguita al verificarsi di uno specifico evento.

Transazioni

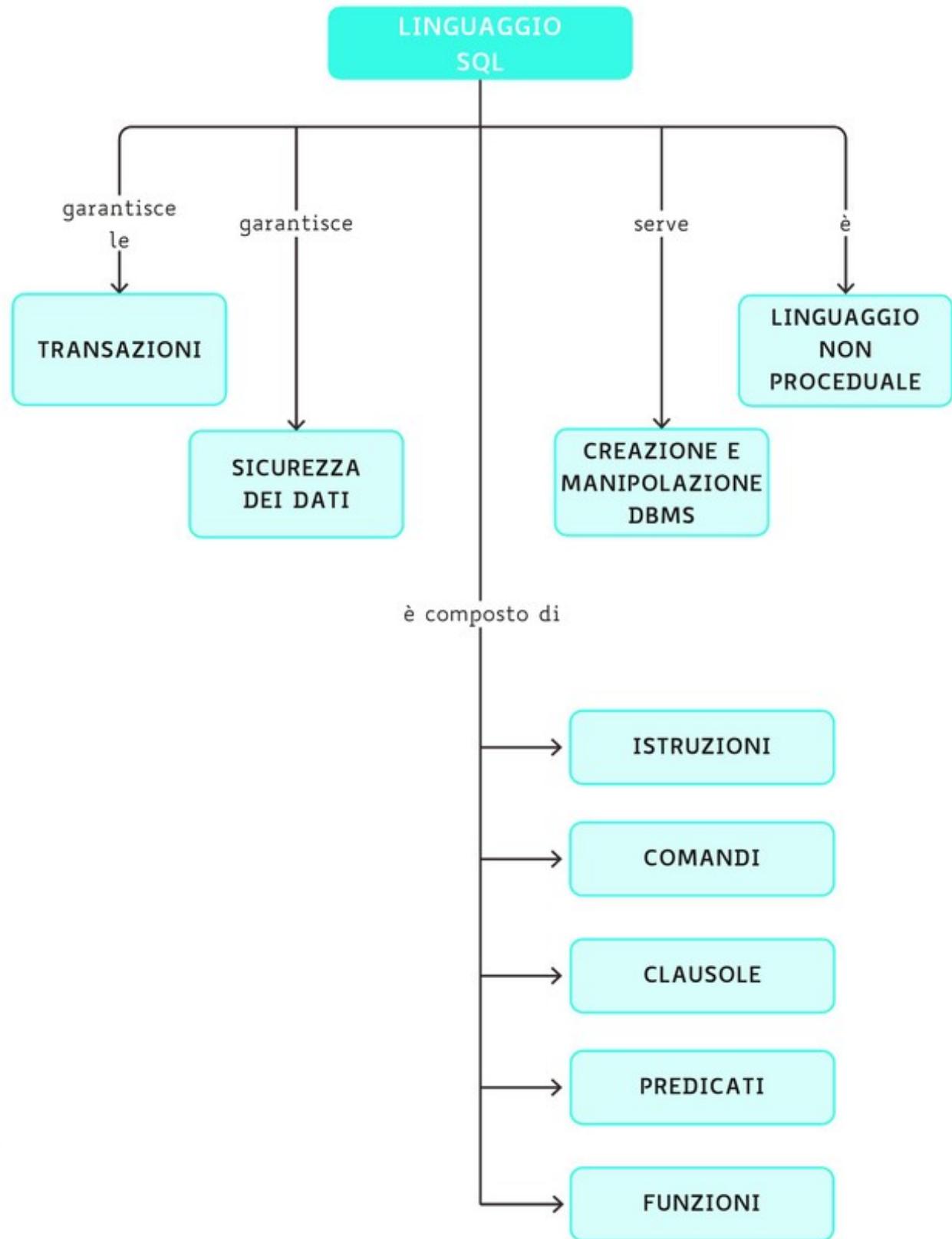
Le transazioni sono un **insieme di istruzioni** che devono essere eseguite senza interruzioni e vengono quindi trattate come se fossero una **singola unità**.

Se si verifica un'**interruzione** della transazione e solo una parte di essa è stata eseguita, il comando ROLLBACK permette di **annullare le modifiche** e di riportare la base di dati in una situazione di integrità.

MAPPA CONCETTUALE



MAPPA
MODIFICABILE



TEST



TEST

Svolgi il test interattivo

Vero o falso?

1. La clausola IN è equivalente a più condizioni OR. V F
2. La clausola HAVING si usa con la clausola GROUP BY. V F
3. Con l'istruzione DELETE elimino uno o più record. V F
4. La clausola ORDER BY serve per ordinare una tabella del database. V F
5. Con l'istruzione ALTER TABLE si modifica il contenuto di una tabella. V F

Scelta multipla (una sola è la risposta esatta)

6. Per sapere quanti sono i record che rispecchiano un criterio si usa la funzione:
 A SUM.
 B MAX.
 C COUNT.
 D AVG. B MAX.
 C COUNT.
 D AVG.
7. GROUP BY serve per:
 A ordinare i record in modo crescente.
 B raggruppare i record in base a valori uguali.
 C selezionare i record in base a valori uguali.
 D ordinare i record in modo decrescente. A ottenere una nuova tabella da una già esistente.
 B inserire e modificare dati.
 C ordinare dati.
 D visualizzare dati.
8. Per sapere il totale dei valori di un gruppo di record occorre usare la funzione:
 A SUM. 9. Un'istruzione SELECT non può essere usata per:
 A ordinare i record in modo crescente.
 B raggruppare i record in base a valori uguali.
 C selezionare i record in base a valori uguali.
 D ordinare i record in modo decrescente.
10. L'istruzione INSERT permette di inserire:
 A una nuova riga in una tabella già esistente.
 B una nuova tabella in un database creato precedentemente.
 C un nuovo campo in una tabella.
 D un campo chiave. 10. L'istruzione INSERT permette di inserire:
 A una nuova riga in una tabella già esistente.
 B una nuova tabella in un database creato precedentemente.
 C un nuovo campo in una tabella.
 D un campo chiave.

PREPARATI PER IL COLLOQUIO ORALE

1. Quali sono le operazioni che permettono di creare e modificare un database relazionale? [vedi Lez. 1 e 2]
2. Quali sono le caratteristiche dell'istruzione SELECT? [vedi Lez. 4 e 5]
3. Di quali funzioni dispone l'SQL? [vedi Lez. 8]
4. Quando si fanno dei raggruppamenti sui dati? [vedi Lez. 9]
5. In quali casi si utilizzano le interrogazioni annidate? [vedi Lez. 10 e 11]
6. Com'è possibile definire una vista con il linguaggio SQL? [vedi Lez. 13]
7. Quando si usa l'operazione di unione di tabelle? [vedi Lez. 12]
8. Quali garanzie fornisce una transazione? [vedi Lez. 15]



CLIL – IN ENGLISH, PLEASE

**AUDIO**Ascolta la pronuncia
del testo**ABSTRACT****The SQL language**

Structured Query Language (SQL) is a query language designed to read, edit, and manage data stored in a relational database, to create and modify database schemas and to create and manage tools to control and access data. A table can be created using the CREATE TABLE statement.

The SELECT statement is the most complex and frequently used statement for performing all types of queries. SQL allows the user to make nested queries – query within other queries. A subquery returns the data (tables or individual data) necessary for the execution of queries at a higher level.

Views are “virtual” tables; they do not contain the table data but formulas for extracting the data. A view is created by using the CREATE VIEW statement. Triggers and transactions are defined to maintain the integrity of the database. A trigger is a procedure that is performed when an event occurs. A transaction is a sequence of operations that may end with success or failure; in case of failure, it will roll back to the state before the transaction.

GLOSSARY**Create table:** creates a new table.**Select from:** extracts data from a table.**Create view:** creates a view.**Trigger:** a procedure that is performed automatically when a particular event occurs.**Transaction:** a sequence of operations that may end with success or failure; in case of failure, it will roll back to the state before the transaction.**Rollback:** an operation that allows the user to repair the database by returning it to a previous version after an error.**Commit:** an operation that allows the user to confirm the transaction. All changes previously made to the data are stored.**EXERCISES****True or false?**

1. A CREATE TABLE statement is used to define the structure of a table. T F
2. The ORDER BY clause is used to sort a table in the database. T F
3. It is not possible to nest SELECT statements. T F
4. You can use the SELECT statement to join two tables. T F
5. You can define views in SQL. T F

Multiple choice

6. A SELECT statement cannot be used to:
 - A get a new table from an existing one.
 - B enter and edit data.
 - C sort data.
 - D display data.
7. The SQL language cannot be used to:
 - A define the structure of a table.
 - B sort the data in a table.
 - C define the access mode of a given user.
 - D define how to access a file.

**GLOSSARIO
CLIL**

**COMPETENZE DISCIPLINARI**

- Identificare e applicare le metodologie e le tecniche della gestione per progetti.
- Redigere relazioni tecniche e documentare le attività individuali e di gruppo relative a situazioni professionali.
- Interpretare i sistemi aziendali nei loro modelli, processi e flussi informativi con riferimento alle differenti tipologie di imprese.

COMPETENZE DEL XXI SECOLO**Competenze trasversali****PENSIERO CRITICO**

Saper analizzare e valutare situazioni in modo da impiegare informazioni e idee per formulare risposte e soluzioni.

COLLABORAZIONE

Saper lavorare in gruppo in vista di un obiettivo comune, prevedendo ed eventualmente gestendo conflitti.

Qualità caratteriali**CURIOSITÀ**

Inclinazione a porre domande con una mentalità aperta.

OBIETTIVI FORMATIVI

- Definire lo schema relazionale.
- Realizzare le query in SQL.
- Esporre i risultati ottenuti alla classe.

TEMPI

- Definire lo schema: 30 minuti.
- Realizzare le query richieste: 1 ora.

STRUMENTI

- Libro di testo.
- Dispositivo connesso a Internet.
- Foglio di carta.
- Software per la realizzazione di database e codifica SQL, PowerPoint.
- Proiettore collegato al computer in classe o in laboratorio.

IL TEMA

Di seguito è riportato un estratto della prima parte della **prova di informatica** dell'Esame di Stato del 2015 per l'Istituto Tecnico, settore Tecnologico, indirizzo Informatica e Telecomunicazioni.

Abbiamo presentato il testo del tema al termine delle unità 2 e 3. Abbiamo preso in considerazione i punti 1 e 2 al termine dell'unità 2 e il punto 3 al termine dell'unità 3.

In questa unità prendiamo in considerazione i punti 4 e 5.

ESTRARRE ELENCHI ED EFFETTUARE CALCOLI

[...] (Puoi vedere il testo del tema alla p. 66).

*Il candidato, fatte le opportune ipotesi aggiuntive, sviluppi:
[...]*

4. la definizione in linguaggio SQL di un sottoinsieme delle relazioni della base di dati in cui siano presenti alcune di quelle che contengono vincoli di integrità referenziale e/o vincoli di dominio, laddove presenti;
5. le seguenti interrogazioni espresse in linguaggio SQL:
 - a. elenco degli eventi già svolti, in ordine alfabetico di provincia;
 - b. elenco dei membri che non hanno mai inserito un commento;
 - c. per ogni evento il voto medio ottenuto in ordine di categoria e titolo;
 - d. i dati dell'utente che ha registrato il maggior numero di eventi.

I nuclei tematici fondamentali e gli obiettivi

Fra i nuclei tematici fondamentali e gli obiettivi cui si riferiscono le prove d'esame (vedi la sezione finale "La preparazione alla seconda prova scritta dell'esame di Stato"), questo estratto si riferisce ai seguenti.

Nuclei tematici fondamentali	Obiettivi
<ul style="list-style-type: none"> Linguaggio per basi di dati: creazione, manipolazione e interrogazione di una base di dati. 	<ul style="list-style-type: none"> Affrontare situazioni problematiche, utilizzando adeguate strategie cognitive e procedure operative orientate alla progettazione di soluzioni informatiche. Scegliere sistemi e strumenti idonei al contesto proposto, in base alle loro caratteristiche funzionali.

Svolgimento

Per la risoluzione del punto 4 utilizziamo l'istruzione CREATE TABLE per creare le tabelle Evento ed Artista. Nella tabella Evento vi sono vincoli di integrità referenziale, come è richiesto nel testo, e in Artista il nome d'arte è un campo opzionale.

```
CREATE TABLE Evento (
    idE      VARCHAR(8)  NOT NULL ,
    titolo   VARCHAR(20) NOT NULL ,
    luogo    VARCHAR(15) NOT NULL ,
    data     DATE        NOT NULL ,
    provincia VARCHAR(4) NOT NULL ,
    cod_utente VARCHAR(12) NOT NULL ,
    cod_categoria VARCHAR(8) NOT NULL ,
    PRIMARY KEY (idE),
    FOREIGN KEY cod_utente REFERENCES utente(nickname),
    FOREIGN KEY cod_categoria REFERENCES categoria(idc)
) ENGINE = InnoDB;
```

```
CREATE TABLE Artista (
    idA      VARCHAR(10) NOT NULL ,
    nome    VARCHAR(25) NOT NULL ,
    cognome VARCHAR(25) NOT NULL ,
    nome_arte` VARCHAR(25) NULL ,
    PRIMARY KEY (idA)) ENGINE = InnoDB;
```

Passiamo ora ad analizzare le richieste del punto 5:

a. elenco degli eventi già svolti, in ordine alfabetico di provincia.

I dati che interessano sono contenuti nella tabella Evento. Dobbiamo prendere solo le righe della tabella che hanno la data minore di quella attuale (la data attuale è restituita dalla funzione now()) e ordinarli secondo il campo provincia.

```
SELECT *
FROM Evento
WHERE data < now()
ORDER BY provincia;
```

b. elenco dei membri che non hanno mai inserito un commento.

Per ottenere i dati degli utenti che non hanno inserito un commento, usiamo due query annidate.

Con quella più interna selezioniamo gli utenti che hanno inserito commenti, con quella più esterna prendiamo nickname, nome e cognome degli utenti che non sono presenti nei risultati della query precedente.

```
SELECT nickname, nome, cognome
FROM Utente
WHERE nickname NOT IN (SELECT cod_utente FROM Commenta);
```

c. per ogni evento il voto medio ottenuto in ordine di categoria e titolo.

Selezioniamo titolo e luogo dalla tabella Evento e calcoliamo la media dei voti (AVG) assegnati a ciascun evento. Abbiamo bisogno di raggruppare i dati per titolo e visualizzare i risultati in ordine di categoria e titolo e quindi, poiché le tabelle utilizzate sono due, nella clausola WHERE inseriamo la condizione per la congiunzione.

```
SELECT titolo, luogo, AVG(voto) AS voto_medio
FROM Evento, Commenta
WHERE idE = Commenta.cod_evento
GROUP BY titolo
ORDER BY cod_categoria, titolo;
```

d. i dati dell'utente che ha registrato il maggior numero di eventi.

Questa query è abbastanza complessa; per realizzarla creiamo la vista T, che conterrà per ogni utente il codice e il numero di commenti inseriti.

A questo punto, utilizzando T con tre query annidate, calcoliamo il massimo dei valori e selezioniamo nickname, nome e cognome dell'utente che ha inserito il massimo numero di commenti.

```
CREATE VIEW T (cod_utente, Ncommenti) AS
SELECT cod_utente, count(*)
FROM Commenta
GROUP BY cod_utente
```

```
SELECT nickname, nome, cognome
FROM Utente WHERE nickname = (SELECT cod_utente
FROM T
WHERE Ncommenti = (SELECT MAX(Ncommenti) FROM T))
```

Le query sono state realizzate partendo dal seguente schema logico definito nel modulo precedente:

Evento (idE, titolo, luogo, data, provincia, cod_utente, cod_categoria)

Artista (idA, nome, cognome, nome_d'arte)

Categoria (idC, descrizione)

Utente (nickname, e-mail, nome, cognome, prov, login, psw)

Partecipa (cod_artista, cod_evento)

Commenta (voto, data, commento, cod_evento, cod_utente)

COMPITI DI REALTÀ

Dopo aver confrontato la risoluzione del tema d'esame con la tua soluzione, in gruppo svolgi le seguenti attività.

- COLLABORAZIONE** Effettuate una ricerca in Internet sul tema del **monitoraggio dell'attività** degli utenti delle community. Si tratta di un argomento interessante sotto molti aspetti, tra cui quello della profilazione a fini commerciali.
Facebook, per esempio, fornisce uno strumento statistico sull'attività dei **gruppi**, a disposizione degli amministratori del gruppo (<https://tinyurl.com/y9mjgrwu>).
- PENSIERO CRITICO** Individuate quelle che, secondo voi, sono le potenzialità della **profilazione** attraverso il monitoraggio dell'attività degli utenti nelle community online.
- COLLABORAZIONE** Raccogliete i risultati della vostra ricerca e della vostra discussione in una presentazione in PowerPoint (o in un altro strumento di presentazione), formata al massimo da cinque slide.
- COMUNICAZIONE** In classe, con la supervisione dell'insegnante, condividete la presentazione con i compagni: confrontate le tematiche emerse e discutetele.
- PENSIERO CRITICO** Infine in classe discutete con i compagni lo svolgimento presentato per risolvere i quesiti del tema d'esame; poi proponete eventuali modifiche, che vi sembrino più adeguate.

AUTOVALUTAZIONE

Al termine delle attività rifletti sull'esperienza e completa la tabella di autovalutazione.

Attività	LIVELLO		
	Base	Intermedio	Avanzato
Mi sono orientato correttamente nella risoluzione del tema proposto all'esame di Stato?	Ho seguito la risoluzione proposta, ma non sarei in grado di realizzarla in autonomia, se non in modo poco dettagliato o sommario. <input type="checkbox"/>	Ho seguito la risoluzione proposta, e sarei in grado di realizzarla da solo anche se avrei tralasciato alcuni particolari. <input type="checkbox"/>	Ho seguito la risoluzione proposta e sarei in grado di realizzare questa soluzione ma anche di ipotizzare alternative. <input type="checkbox"/>
Ho interagito con i membri del mio gruppo in modo proficuo e costruttivo?	Ho lavorato principalmente da solo. <input type="checkbox"/>	Ho collaborato con i compagni in tutte le attività. <input type="checkbox"/>	Ho collaborato con i compagni e in alcuni casi ho guidato il lavoro del gruppo. <input type="checkbox"/>
Ho partecipato attivamente al dibattito in classe?	Non sono riuscito a collaborare alle attività del mio gruppo. <input type="checkbox"/>	Ho partecipato al dibattito ma il mio contributo è stato marginale. <input type="checkbox"/>	Ho partecipato al dibattito e il mio contributo è stato determinante. <input type="checkbox"/>
Sono in grado di usare gli strumenti informatici?	Ho alcune difficoltà nell'uso di PowerPoint. <input type="checkbox"/>	Uso correttamente PowerPoint ma in modo amatoriale. <input type="checkbox"/>	Uso PowerPoint da utente esperto. <input type="checkbox"/>

5

Programmare in rete



ESERCIZI COMMENTATI

Segui la risoluzione passo passo



LABORATORI CASE STUDIES

Approfondisci con i casi pratici



PREREQUISITI

- Conoscere le caratteristiche del modello Client/Server.
- Conoscere gli elementi di base di HTML e di VB.NET.

PER COMINCIARE

1. Il tag `<head>` serve per definire:
 - A il corpo del documento
 - B l'intestazione del documento
 - C il colore di sfondo del documento
 - D il tipo di carattere
2. Per creare una lista ordinata si usa il tag:
 - A `<tr>`
 - B ``
 - C `<col>`
 - D ``
3. Per inserire un'immagine si usa il tag:
 - A ``
 - B ``
 - C ``
 - D ``
4. Il link `` serve per:
 - A inserire un'immagine
 - B inserire un link interno
 - C inserire un link esterno
 - D visualizzare la parola fine
5. Che cosa si intende per architettura Client/Server?



CONOSCENZE

- Conoscere le possibilità di programmazione web.
- Conoscere le caratteristiche della programmazione lato client e lato server.



ABILITÀ

- Saper confrontare i linguaggi di scripting.
- Saper confrontare le caratteristiche delle pagine PHP e ASP.



COMPETENZE

- Sviluppare applicazioni informatiche per reti locali o servizi a distanza.
- Gestire progetti informatici che utilizzano le reti secondo gli standard aziendali.

1. PROGRAMMARE APPLICAZIONI WEB

■ Scenari possibili

Lo scambio di informazioni in Internet è basato fondamentalmente sull'**architettura Client/Server**. Questo significa che i terminali client (che fanno parte della rete) contattano un terminale, solitamente molto potente in termini di capacità di entrata e uscita, che prende il nome di server ed eroga servizi sotto forma di programmi che forniscono dati (fig. 1).



fig. 1 Scambio di informazioni basato sull' architettura Client/Server

In base al tipo di applicazione che si vuole realizzare, un terminale client può diventare server e viceversa. Ciascuna delle due componenti può essere *statica* (cioè non rielaborare le informazioni), oppure *dinamica*; in questo secondo caso, a fronte di una richiesta, attiva un programma in grado di effettuare delle operazioni (per esempio generare le informazioni da trasmettere). I casi possibili sono dunque quattro:

- un client statico che interagisce con un server statico;
- un client dinamico che interagisce con un server statico;
- un client statico che interagisce con un server dinamico;
- un client dinamico che interagisce con un server dinamico.

Vediamo più nel dettaglio i singoli casi.

■ Client statico con server statico

È il caso più semplice. Il client invia una richiesta al server, indicando uno specifico URL (*Uniform Resource Locator*). Il server individua la risorsa richiesta e la invia al client. Se si tratta di un file HTML, il client interpreta il codice e visualizza le informazioni o invia un'altra richiesta al server.

Viene utilizzato tipicamente nei cosiddetti "siti vetrina", in cui vi sono solo pagine statiche o immagini.

Nei casi in cui, invece, si accede al server come deposito di file, lo scopo è il download sul computer di un file (nella maggior parte dei casi una immagine o un filmato da utilizzare localmente).

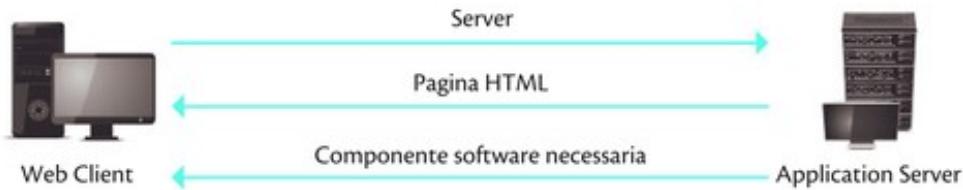
■ Client dinamico con server statico

In questo caso il client svolge una parte dell'elaborazione. Esistono due modi per rendere dinamico un client:

1. il modo più semplice è quello di utilizzare i linguaggi di scripting. In questo caso il codice viene inserito all'interno delle pagine HTML e viene interpretato al momento dell'esecuzione da parte del browser;

2. nel secondo caso, esiste una componente software (per esempio una applet) o un programma JavaScript (come vedremo nelle lezioni successive) che risiede sul server, che viene automaticamente scaricata sul client (quando necessaria) ed eseguita localmente ([fig. 2](#)). In genere vengono realizzate utilizzando un ambiente di sviluppo vero e proprio, sono scritte in un linguaggio di programmazione e quindi compilate. Possono essere utilizzate all'interno di varie pagine HTML. A fronte di opportuni parametri forniti in input vengono forniti i servizi o restituiti i risultati.

[fig. 2](#) Flusso delle informazioni tra client e server

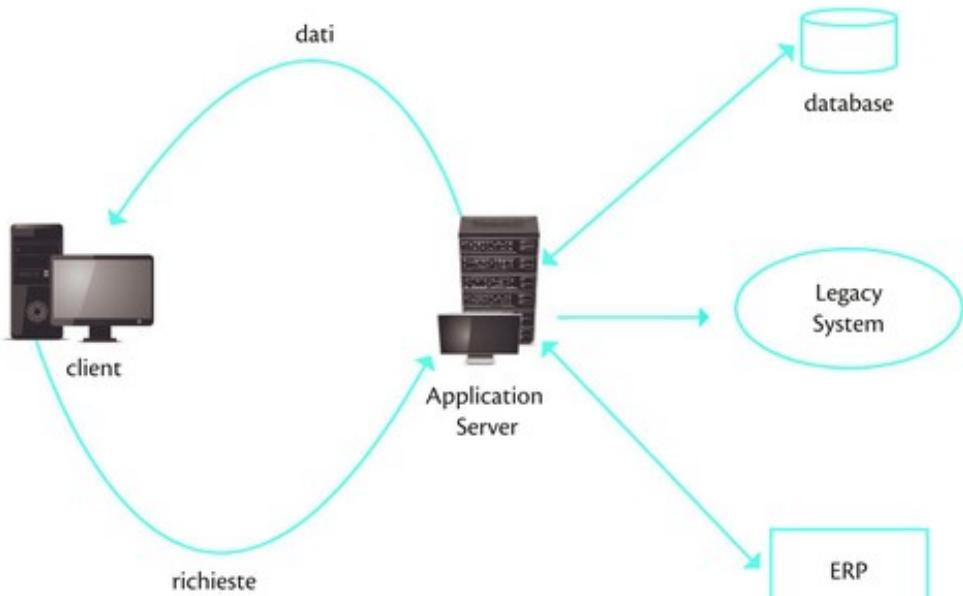


■ Client statico con server dinamico

Se il server è dinamico significa che, a fronte di una richiesta, è in grado di inviare al client una risposta che varia in funzione dei parametri inseriti dall'utente.

Per recuperare l'informazione, il server può eseguire direttamente una query SQL, oppure attivare un applicativo tramite componenti software che fungono da middleware ([fig. 3](#)).

[fig. 3](#) Middleware



LO SAI CHE

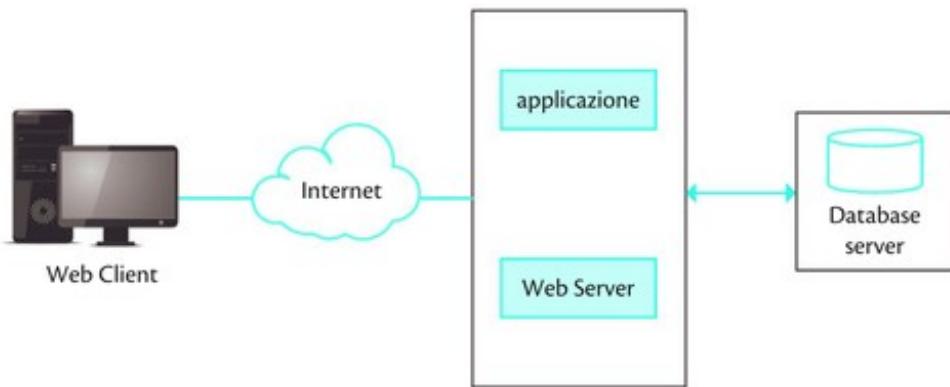
Legacy System

Si tratta dei vecchi sistemi aziendali ancora in funzione con cui le nuove applicazioni si devono interfacciare.

In ambito Internet, il server su cui vengono eseguite queste applicazioni prende il nome di Application Server.

In questo scenario possono esistere più server distinti. Il Web Server (*front-end server*) si occupa di colloquiare da un lato con i client e dall'altro con gli Application Server (*middleware*) per recuperare le informazioni, eventualmente richiedendole a un ulteriore nodo (*back-end server*), come è schematizzato nella [fig. 4](#).

fig. 4 Application Server



■ Client dinamico con server dinamico

Unisce le potenzialità viste in precedenza: permette di migliorare l'integrazione tra utente e sistema, consentendo un accesso adeguato ai dati e alle applicazioni aziendali. È il modello attualmente più adottato per lo sviluppo di applicazioni aziendali basate sugli standard Internet (*Intranet* ed *Extranet*). Per realizzare la dinamicità dei client e dei server, è necessario scrivere programmi: si va dalla realizzazione di semplici **script** interpretabili dai browser, fino alla creazione di interi programmi compilati. Alcune di queste tecniche sono utilizzabili solo dal lato server, altre servono solo per i client, mentre vi sono tecniche che possono essere usate su entrambi i nodi della comunicazione.

■ Pagine statiche

Le pagine **statiche** (quelle che in genere si riconoscono dall'estensione .html o .htm) sono dei **file in codice HTML**, che descrivono minuziosamente testi da impaginare, grafica e immagini.

Quando l'utente di un sito visita una pagina, ciò che avviene è che il server su cui risiede il sito invia al browser (il programma che viene utilizzato per navigare, per esempio Google Chrome) il file HTML; il browser sa decodificare il file, e quindi mostra i contenuti della pagina sullo schermo dell'utente. In questo caso il Web Server si comporta come un hard disk remoto: controlla se la pagina esiste e la fornisce così com'è.

Questo è il sistema originario del web: gli utenti erano pochi ed erano poche anche le informazioni veicolate, Internet serviva come una enorme rete locale, in cui pescare i documenti desiderati (con solo alcuni vantaggi derivanti dal sistema di hyperlink) (fig. 5).

fig. 5 Pagine web statiche



■ Pagine dinamiche

Le pagine **dinamiche**, invece di contenere il codice HTML (o meglio invece di contenere solo quello) ospitano programmi per il server. Il server li esegue e quindi scrive il codice HTML (che non è preesistente come nella pagina statica) da inviare al browser.

In altre parole, il contenuto della pagina non è deciso a priori, ma può variare in base a condizioni di vario genere.

ESEMPIO

L'esempio più chiaro e noto a tutti è quello dei motori di ricerca: il server del motore di ricerca (per esempio quello di Google) non contiene già la pagina che mostra tutte le risorse associate alle ricerche degli utenti, ma la crea volta per volta nel momento in cui un utente immette i termini per la ricerca; solo dopo che il server ha eseguito la sua ricerca sulla parola chiave indicata viene creato il codice per presentare i risultati (fig. 6).



fig. 6 Pagine web dinamiche

Una pagina dinamica ha di solito un'estensione diversa (.php, .jsp, .cfm, .aspx) e spesso anche dei parametri dopo un punto di domanda (per esempio ?sezione=3&pagina=5).

La pagina ha in genere una parte del contenuto fissa, ma anche una parte scritta in un linguaggio di scripting che non viene inviata al browser così com'è, ma viene prima eseguita dal Web Server, che fa delle operazioni, da molto semplici a molto complesse, e invia infine al browser la pagina elaborata.

Di fatto, a questo punto (quando è sullo schermo) la pagina è diventata a tutti gli effetti statica, insieme ai comandi HTML. E infatti può essere salvata, facendone in un certo senso la fotografia, ma perdendone tutte le caratteristiche dinamiche.

Si pensi, per esempio, a un sistema di news: una volta salvata, la pagina può essere riaperta, ma l'ultima news più recente sarà sempre la stessa, mentre online la news si aggiorna continuamente.

FISSA LE CONOSCENZE

1. Quale funzione ha il middleware?
2. Quali caratteristiche ha una pagina dinamica?
3. Cosa si intende per client dinamico?

2. PROGRAMMARE LATO CLIENT

■ Il client dinamico

Come descritto nella precedente Lezione, si dice che il client è dinamico quando, prima di mostrare una pagina all'utente, rielabora localmente le informazioni che provengono dal server.

Quando il client è dinamico, possono presentarsi due differenti modalità di lavoro. Il client può:

- eseguire un programma già compilato che risiede sul server, dopo averlo scaricato (applet, plug-in);
- eseguire un programma scritto in un linguaggio di scripting; in questo caso il codice dello script viene spedito dal server insieme al codice HTML.

■ Applet

Si tratta di particolari programmi Java che vengono inseriti nelle pagine HTML, ma che non possono essere eseguiti indipendentemente dal programma che li ha richiamati.

Una applet è richiamata con il tag `<applet code="Bubbles.class">` seguito dai parametri che indicano le dimensioni del rettangolo che la applet occupa all'interno della pagina che la ospita:

```
<applet code="NomeApplet" width="350" height="350">.
```

Quando un browser incontra tali tag HTML, carica in memoria i programmi e li esegue (fig. 7).

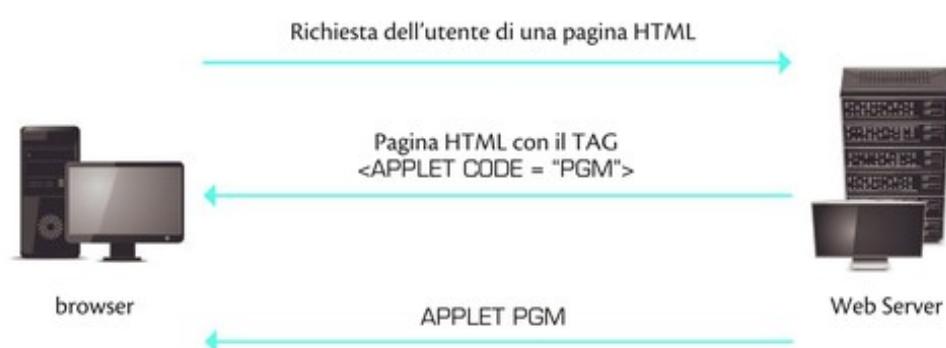


fig. 7 Applet e Client/Server

Le applet Java sono sicure, perché non possono accedere alle risorse locali del computer (per esempio l'hard disk del client). Esempi di applet sono Adobe Flash e Windows Media Player.

Plug-in

I plug-in sono componenti software che permettono di estendere le funzionalità del browser, in modo da consentirgli di interpretare dati in un formato particolare (non HTML).

Quando il browser effettua il download di una pagina con dati di un formato che non è in grado di interpretare, verifica se possiede un plug-in in grado di gestirli.

Se necessario, il plug-in può essere scaricato dal server e automaticamente installato dopo il download. Il plug-in può operare all'interno di una pagina HTML al fine di visualizzare dati in formato proprietario: per esempio, un player AVI/MPEG consente di visualizzare filmati codificati in questi formati. Spesso un plug-in funziona come applicativo separato, utilizzato per la visualizzazione di un intero documento (per esempio Adobe Acrobat Viewer). Può anche essere eseguito in background (per esempio MIDI player). Alcune funzionalità tipiche sono: la visualizzazione di formati grafici (CAD), l'esecuzione di presentazioni interattive (Flash), l'esecuzione di codice Java (applet).

Linguaggi di scripting

Si tratta di porzioni di codice inserite in pagine HTML. La funzione di queste piccole applicazioni consiste nell'introdurre estensioni all'interfaccia di una pagina web o del browser. In questo modo si aumentano le potenzialità interattive di una pagina web senza ricorrere allo sviluppo di plug-in o di applet Java, attività che richiede la competenza di un programmatore. Gli script possono essere eseguiti sia a livello server, sia a livello client. Quando gli script sono eseguiti a livello client non possono accedere alle risorse locali (sono quindi sicuri). I principali linguaggi di scripting sono **JavaScript** e **Visual Basic Script (VBScript)**.

JavaScript

Nonostante il nome, il linguaggio JavaScript non ha niente a che fare con il linguaggio Java. Il codice viene inserito direttamente nelle pagine HTML preceduto dal tag <SCRIPT LANGUAGE = "JavaScript">. Il codice è normalmente composto da una serie di function, che vengono richiamate in corrispondenza di eventi che occorrono nella pagina.

Viene usato principalmente per realizzare form per l'immissione di dati, realizzare calcoli su tabelle e per gestire la navigazione sul web. Come le applet Java, non ha accesso alle risorse locali.

Visual Basic Script (VBScript)

Il linguaggio VBScript deriva dal Visual Basic. Come per JavaScript, il codice è inserito in pagine HTML dopo il tag <SCRIPT LANGUAGE = "VBScript">. Le funzionalità offerte sono praticamente le stesse. La principale differenza sta nella portabilità (decisamente inferiore a JavaScript, in quanto utilizzabile solo con Internet Explorer) e nella fase di apprendimento del linguaggio (decisamente più facile di JavaScript).

LO SAI CHE

VBScript deprecato
Dalla versione 11 di Internet Explorer VBScript non viene più supportato, e se ne sconsiglia l'uso (viene cioè deprecato) proprio per la sua poca portabilità. I vecchi script possono però continuare a funzionare tramite l'uso di appositi comandi di interfaccia in modo da garantire il funzionamento dei vecchi sistemi legacy.

FISSA LE CONOSCENZE

1. Che cosa è una applet?
2. A che cosa servono i plug-in?
3. Quali sono i linguaggi di scripting?

3. PROGRAMMARE LATO SERVER

■ Il server dinamico

Si dice che il server è dinamico quando le pagine o le informazioni che vanno inviate al client non sono immediatamente disponibili, ma devono essere reperite sul server (o su altri sistemi) o create al momento.

La programmazione di componenti software per web Application lato server può sfruttare diverse tecnologie. Le più comuni sono:

- CGI (*Common Gateway Interface*);
- Java servlet;
- linguaggi di scripting quali: Active Server Pages (ASP), Java Server Pages (JSP) e PHP (*Hypertext Processor*).

Interfaccia CGI

La CGI (*Common Gateway Interface*) è un'interfaccia che definisce il modo in cui il server e alcuni programmi (programmi gateway) comunicano tra loro. Lo scambio di informazioni avviene mediante variabili d'ambiente speciali, argomenti della riga di comando o, nel caso più frequente, mediante i parametri di un modulo da compilare (form HTML) (fig. 8).

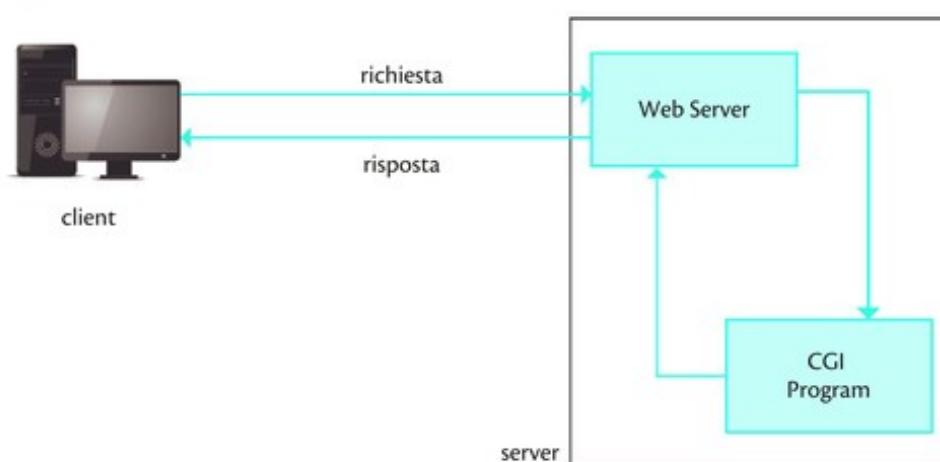


fig. 8 Tecnologia CGI

Componenti Java servlet

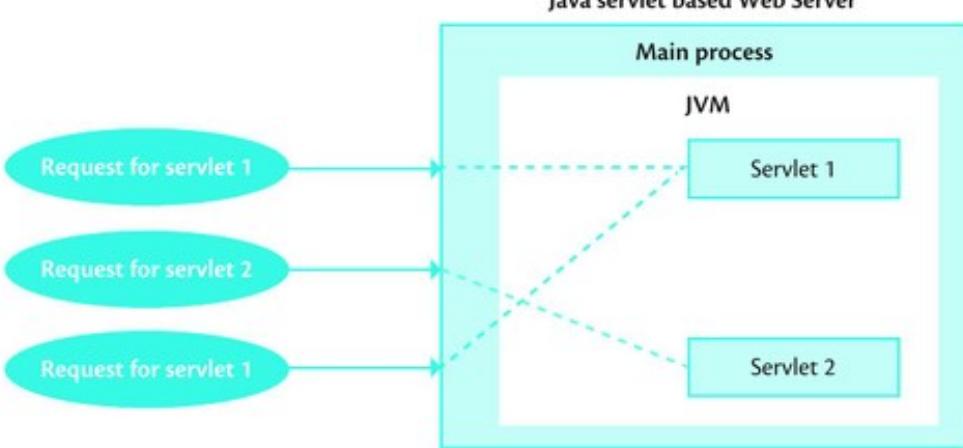
Java è un linguaggio realizzato da Sun Microsystems, che richiama per molti aspetti il C++. Rappresenta sia un linguaggio sia una piattaforma per lo sviluppo di applicazioni a oggetti distribuite.

Le **servlet** (fig. 9) sono componenti Java che vengono eseguite sulla piattaforma server (Application Server) e possono fornire al client dei contenuti, espressi in formato HTML, costruiti dinamicamente in fase di esecuzione.

Il tempo necessario al caricamento e all'inizializzazione di una servlet incide una sola volta, alla prima richiesta.

La servlet diventa quindi parte integrante dello stesso Application Server. Quando la sessione viene chiusa, anche il *thread* legato alla servlet viene chiuso.

fig. 9 Servlet



L'interfaccia CGI definisce solo il protocollo di interazione tra server e applicazione e quindi non stabilisce in quale linguaggio quest'ultima dovrà essere scritta. Qualsiasi applicazione che rispetta tale protocollo potrà diventare una applicazione CGI o, meglio, un servizio web.

Pagine JSP

Le Java Server Pages (JSP) sono pagine HTML contenenti codice Java che viene eseguito sulla piattaforma server.

La **prima volta** che la pagina è richiamata, viene tradotto il codice sorgente Java. La pagina viene quindi compilata sotto forma di servlet e memorizzata su disco, e poi mandata in esecuzione. Ai successivi accessi alla pagina, la componente servlet viene direttamente eseguita perché è già compilata e memorizzata.

```
<HTML>
<HEAD> <TITLE>
    Printing numbers
</TITLE> </HEAD>
<BODY BGCOLOR = #FFFFFF>
<java>
    for (int i = 0; i < 5; i++)
        out.println ("<BR>" + i); </java>
<BR>
</BODY>
</HTML>
```

Pagine ASP

Le Active Server Pages (ASP) forniscono un meccanismo con cui è possibile inserire all'interno di una pagina web il codice, che poi viene eseguito sul server a ogni accesso alla pagina (fig. 10).

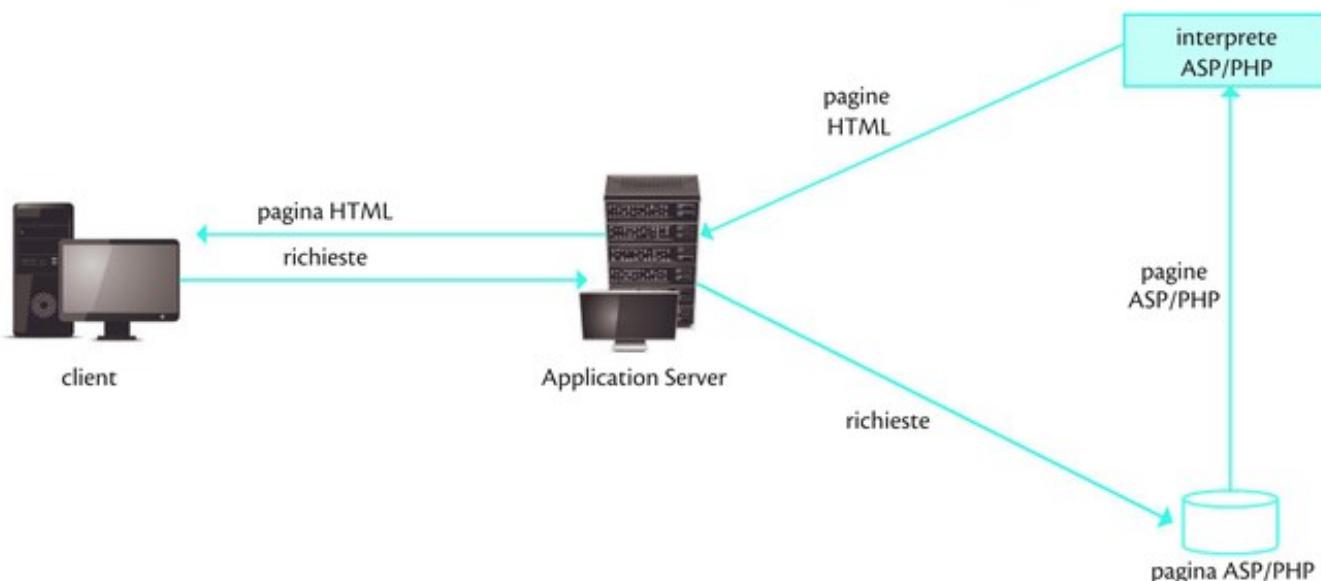
La pagina inviata al browser viene quindi **generata dinamicamente** in base a specifici parametri forniti in input.

Con le ASP è possibile comunicare con altre componenti applicative, accedere ai database e alle applicazioni legacy.

La tecnologia ASP è disponibile solo su piattaforme Microsoft, ma la pagina generata è visualizzabile su qualsiasi browser. Il codice è interpretato

da una componente (ASP.DLL), che opera parallelamente al Web Server ed è in grado di gestire più richieste contemporaneamente (*multithread*).

fig. 10 Pagine dinamiche



Pagine PHP

Le pagine PHP, dal punto di vista funzionale, sono simili alle pagine ASP. Sono pagine HTML che contengono codice scritto in linguaggio PHP (*Hypertext Preprocessor*; l'acronimo ha origine da *Personal Home Page*, perché questo linguaggio fu concepito per la programmazione di pagine web dinamiche).

Anche in questo caso la pagina viene prelevata sul server, rielaborata e integrata con dati presi dal database in base a specifici parametri.

Una volta preparata, la pagina viene inviata al browser che la potrà visualizzare come se fosse una pagina statica.

PHP è un **linguaggio completo di scripting**, sofisticato e flessibile; può girare praticamente su qualsiasi Web Server, su qualsiasi sistema operativo, e consente di interagire praticamente con qualsiasi tipo di database (MySQL, Access, SQL Server, Oracle e altri).

PHP può essere agevolmente utilizzato per svariati tipi di progetti, dalla semplice home page dinamica fino al grande portale o al sito di e-commerce.

Quando un comando dello script richiede il reperimento di dati da un database, il Web Server preleva i dati dal database e li inserisce nei comandi HTML all'interno del modulo PHP, producendo un documento HTML formattato (fig.10).

FISSA LE CONOSCENZE

1. Che cosa sono le pagine ASP?
2. Che rapporto c'è tra pagine JSP e le servlet?
3. Che cosa si intende per CGI?
4. Come funziona il meccanismo delle pagine ASP e PHP?



RIPASSIAMO INSIEME

Programmare applicazioni web

L'architettura **Client/Server** è il modello su cui si basa lo scambio delle informazioni in rete. Con il termine **server** si intende una macchina che fornisce servizi, per **client** si intende invece una macchina che richiede i servizi. Ciascuna di queste due componenti può essere **statica** (non rielabora le informazioni), oppure **dinamica**, quando è in grado di elaborare informazioni attivando i programmi necessari.

Programmazione lato client

Le **tecniche dal lato client** sono caratterizzate dal fatto che l'esecuzione e l'interpretazione delle istruzioni avvengono **in locale**, sul computer che effettua la richiesta al server. Il vantaggio di un'esecuzione lato client consiste nel liberare il server di una parte di lavoro che viene passata al client. Le **applet Java** sono programmi scritti

in linguaggio Java che, collocati all'interno di pagine web, possono essere eseguiti da un web browser.

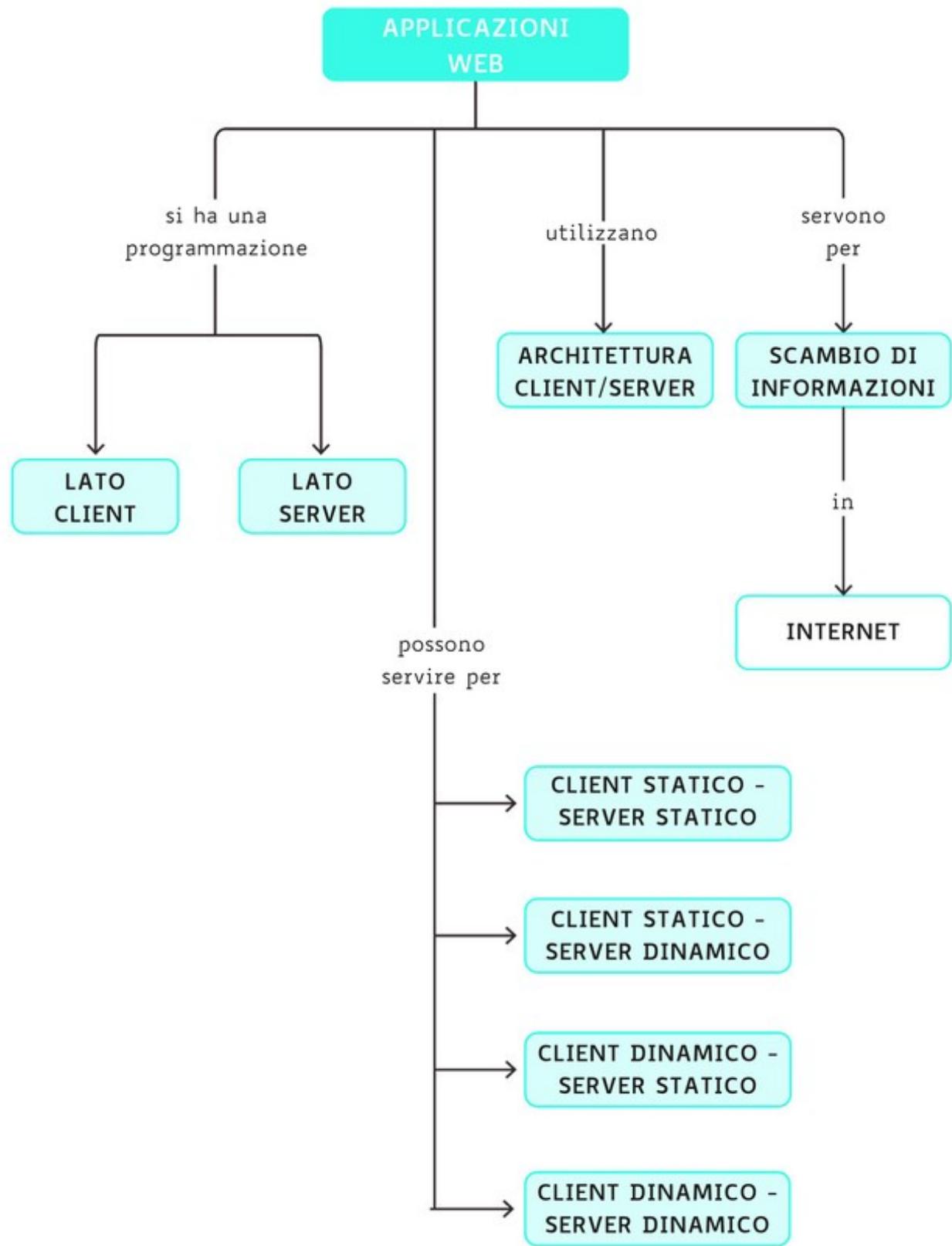
Il **plug-in** è un programma non autonomo che interagisce con un altro programma per ampliarne le funzioni.

Il **linguaggio di scripting** è un linguaggio di programmazione interpretato (cioè, che non viene compilato).

Programmazione lato server

La **programmazione lato server** comprende l'insieme delle tecniche che consentono di realizzare **siti dinamici** e applicazioni web **interattive** con la possibilità di accedere a basi di dati tramite interfacce web. Si tratta di realizzare applicazioni che girano sul server e forniscono dinamicamente informazioni ai client. I principali linguaggi e tecnologie che possono essere utilizzati dal lato server sono: Java servlet, JSP, ASP.NET e PHP.





TEST**TEST**

Svolgi il test interattivo

Vero o falso?

1. Un programma ASP.NET o una pagina PHP sono eseguite dal client. V F
2. L'output di una pagina dinamica è una pagina HTML. V F
3. Javascript è un linguaggio compilato. V F
4. Generalmente un programma Javascript è eseguito dal client. V F
5. Un plug-in non può essere eseguito in modo autonomo. V F
6. Le JSP sono pagine HTML che contengono codice C++. V F
7. Le applet sono programmi eseguiti dal client. V F

Scelta multipla (una sola è la risposta esatta)

8. Una pagina ASP.NET è formata da:
 - A codice HTML.
 - B codice HTML e codice VB.NET.
 - C codice VB.NET.
 - D richiami al database tramite tecnologia.
 9. Un server permette di:
 - A visualizzare una pagina HTML.
 - B eseguire script.
 - C eseguire pagine dinamiche.
 - D eseguire applet.
 10. Il risultato dell'esecuzione di una pagina PHP (o ASP) è:
 - A una pagina HTML.
 - B una applet.
 - C una pagina PHP (o ASP).
 - D nessun risultato.
 11. Uno script JavaScript è:
 - A un programma a sé stante.
 - B un programma Java.
- C una funzione inserita in una pagina HTML.
 D una servlet.
12. Una applet è:
 - A un programma a sé stante.
 - B un programma Java.
 - C una funzione inserita in una pagina HTML.
 - D una servlet.
 13. CGI vuol dire:
 - A Common Gateway Interface.
 - B Common General Interface.
 - C C++ Gateway Interface.
 - D Common Gateway Interaction.
 14. Il server su cui vengono eseguite le applicazioni web prende il nome di:
 - A Middleware.
 - B Application Server.
 - C DB Server.
 - D Monitor Server.

PREPARATI PER IL COLLOQUIO ORALE

1. Che cosa si intende per modalità Client/Server? [vedi lez. 1]
2. Che differenza c'è tra pagine statiche e pagine dinamiche? [vedi lez. 1]
3. Che cosa sono i linguaggi di scripting? [vedi lez. 2]
4. Su che linguaggio sono basate le pagine JSP? [vedi lez. 3]



CLIL – IN ENGLISH, PLEASE



AUDIO
Ascolta la pronuncia
del testo

ABSTRACT

Network programming

The Client/Server model is the model used for the exchange of information on the Internet. The term server refers to a computer that provides services, whereas a client is a computer that requests the services. Each of these two components can be static (does not reprocess the information) or dynamic when it is able to process information by activating the necessary programs.

Client-side technologies are characterized by the fact that the execution and interpretation of statements is carried out locally on the computer making the request to the server. Plug-ins, Java applets, and statements in scripting languages embedded in HTML programs are used.

Server-side programming includes the set of techniques that permit the creation of dynamic web sites and interactive Web

Applications. The key component of these applications is the ability to access databases through web interfaces. The main languages and technologies that can be used server side are: Java Servlet, JSP, ASP.NET, and PHP.

EXERCISES

True or false?

1. The client cannot process information in the Client/Server model. T F
2. Applets are programs run by the client. T F
3. JavaScript is a scripting language. T F

Multiple choice

4. A server allows the user to:
 - A display an HTML page
 - B execute scripts
 - C execute dynamic pages
 - D run applets
5. JavaScript is:
 - A a standalone program
 - B a Java program
 - C a feature placed onto an HTML page
 - D a servlet
6. An applet is:
 - A a standalone program
 - B a Java program
 - C a Java program placed onto an HTML page
 - D a function of a scripting language

GLOSSARY

Plug-in: is a software component that adds a specific feature to an existing software application.

Applet: small application that performs one specific task that runs inside the client.

JavaScript: is an interpreted computer programming language used to create client-side scripts to interact with the user.

Middleware: software which interfaces between client and the server.

Web Server: program which receives online requests from clients and returns HTML documents.

Script: lines of programming code inserted in the database.

PHP (Personal Home Page tools): programming language used to produce dynamic Internet pages.

ASP.NET: software development technologies for producing dynamic Internet pages.



GLOSSARIO
CLIL

6

PHP e MySQL



LABORATORI CASE STUDIES

Approfondisci con i casi pratici



PREREQUISITI

- Conoscere le caratteristiche del modello Client/Server.
- Conoscere gli elementi di base dei linguaggi di programmazione.
- Conoscere la programmazione con HTML.

PER COMINCIARE

1. Il tag `<head>` serve per definire:

- A il corpo del documento
- B l'intestazione del documento
- C il colore di sfondo del documento
- D il tipo di carattere

2. Una base di dati è:

- A un insieme di programmi
- B un insieme di dati strutturati
- C un insieme di routine per la gestione dei file
- D un insieme di interfacce grafiche

3. Data la tabella `Impiegato` (`Codf`, `DatiAnagrafici`, `Stipendio`) il comando `select avg (stipendio), from Impiegato`, fornisce:

- A l'impiegato che guadagna di più
- B il numero medio di impiegati
- C il totale degli stipendi
- D la media degli stipendi

CONOSCENZE

- Conoscere le possibilità di programmazione web.
- Conoscere le caratteristiche della programmazione lato client e lato server.
- Conoscere la programmazione PHP.
- Conoscere MySQL.

ABILITÀ

- Saper eseguire semplici query su un database MySQL.
- Saper estrarre un insieme di record.
- Saper confrontare le caratteristiche delle pagine HTML e PHP.

COMPETENZE

- Scegliere dispositivi e strumenti per sviluppare applicazioni informatiche per reti locali o servizi a distanza.
- Utilizzare i sistemi informativi aziendali e gli strumenti di comunicazione integrata d'impresa, per realizzare attività comunicative con riferimento a differenti contesti.

1. LINGUAGGIO PHP

La diffusione di PHP ha avuto un notevole incremento dalla fine del secolo scorso a oggi, per due motivi di fondo: il fatto di essere un prodotto open source, quindi disponibile e gratuito, e l'elevato grado di portabilità, in quanto PHP può funzionare su moltissime piattaforme, sia per quanto riguarda i sistemi operativi, sia per quanto riguarda i server web.

■ Il linguaggio PHP

Un file PHP differisce da un documento HTML per la presenza di righe di comando scritte in linguaggio di scripting, insieme ai comandi HTML.

I punti in cui inizia e termina il codice scritto in PHP sono segnalati rispettivamente dai tag: <? e ?> oppure <?PHP e ?>. Tutto il codice racchiuso tra questi simboli viene elaborato dal server web prima che la pagina sia inviata al browser del client.

Per visualizzare il codice da inviare al browser, PHP mette a disposizione due costrutti, che possiamo considerare equivalenti: print() ed echo().

Il codice:

```
<?
    print("Buongiorno a tutti!<br>");
    echo("È una bellissima giornata");
?>
```

visualizza sul browser:

```
Buongiorno a tutti!
È una bellissima giornata
```

I **commenti** svolgono un ruolo fondamentale nella stesura del codice, perché possono facilitare di molto la comprensione dei passaggi. È bene dunque non risparmiare mai un commento quando possibile, anche perché il loro utilizzo non appesantisce l'esecuzione dello script (l'interprete PHP salta tutte le parti che riconosce come commenti), né il trasferimento della pagina al browser (infatti i commenti, essendo contenuti all'interno del codice PHP, fanno parte di ciò che non viene inviato al browser). Il commento è caratterizzato da due barre:

```
// Questo è un commento
```

Variabili

Le **variabili** non devono essere dichiarate, ma il loro nome deve essere preceduto dal simbolo del dollaro (\$).

Per esempio, con l'istruzione:

```
$a = 5;
```

si definisce la variabile a e le si assegna il valore 5.

In fondo all'istruzione c'è il punto e virgola, che deve chiudere tutte le istruzioni PHP.

Le stringhe sono delimitate dagli apici:

```
<? $nome = 'Gigi'?>
```

e la loro **concatenazione** avviene con il carattere punto (.):

```
$nome = 'Gigi';
$stringa1 = 'ciao' . $nome; // $stringa1 vale 'ciao Gigi'
```

Operatori

Gli operatori disponibili sono elencati nella [tabella 1](#).

tabella 1 Operatori PHP

operatore	simbolo
Addizione	+
Moltiplicazione	*
Modulo	%
Uguale a	==
Maggiore di	>
Minore di	<
AND	&&
NOT	!
Sottrazione	-
Divisione	/
Incremento di una unità	++
Decremento di una unità	--
Diverso da	!=
Maggiore o uguale a	>=
Minore o uguale a	<=
OR	
XOR	^

Strutture condizionali

La sintassi dell'**istruzione condizionale completa** è la seguente:

```
<?    If [condizione]
      {
          Blocco Istruzioni1}
      else
      {
          Blocco Istruzioni2}
?>
```

Nel caso di **struttura condizionale a una via**, non è presente il ramo Else.
La struttura di selezione multipla presenta questa sintassi:

```
<? switch (variabile) {  
    case valore1:  
        Blocco Istruzioni1;  
        break;  
    case valore2:  
        Blocco Istruzioni2;  
        break;  
    .....  
    default:  
        Blocco istruzioni_default;}  
?>
```

L'operatore ternario ha la seguente forma:

```
($altezza >= 180) ? $tipo = 'alto': $tipo = 'normale'
```

assegna a \$tipo il valore "alto" se la variabile \$altezza è maggiore o uguale a 180, e "normale" altrimenti.

In pratica si può considerare, questa, una maniera molto sintetica di effettuare una IF.

L'operatore ternario è così chiamato perché è formato da tre espressioni: il valore restituito è quello della seconda o della terza di queste, a seconda che la prima sia vera o falsa.

Strutture di iterazione

Il **ciclo while** permette di ripetere un blocco di istruzioni fino a quando la condizione di ciclo rimane vera:

```
<? while (condizione) {  
    Blocco Istruzioni}  
?>
```

Se il codice indicato nelle parentesi graffe deve essere eseguito almeno una volta, si può utilizzare la seguente struttura:

```
<? do {  
    Blocco Istruzioni1}  
    while (condizione);  
?>
```

Il **ciclo con contatore** può essere scritto con l'istruzione For:

```
<? for (valore iniz. contatore; condiz.; incremento) {  
    Blocco Istruzioni1}  
?>
```

Per esempio, il codice seguente stampa i primi 10 multipli del numero 2:

```
<? for ($mul = 1; $mul <= 10; $mul++) {  
    echo ( " ". $mul*2 ) ;}  
?>
```

Array

Per dichiarare un array, si possono elencare i suoi valori separati da virgolette:

```
$nomi = array('Luca', 'Anna', 'Luisa', 'Marco', 'Giulia')
```

L'array \$nomi ha 5 elementi, numerati da 0 a 4, perché la numerazione degli indici dell'array inizia da 0. Gli elementi dell'array sono identificati dall'indice, che viene indicato tra parentesi quadre.

Per esempio, l'istruzione:

```
print $nomi[2]
```

stampa Luisa.

Per aggiungere un valore all'array, si può utilizzare questa istruzione:

```
$nomi[] = 'Olivia'
```

Con questo codice viene creato un nuovo elemento nell'array \$nomi, con indice 5. Questa sintassi può essere utilizzata anche per definire un array, in alternativa a quella usata prima: infatti, se ipotizziamo che l'array \$nomi non fosse ancora definito, questa istruzione lo avrebbe definito creando l'elemento con indice 0.

In PHP oltre agli array scalari, dove gli elementi sono individuati tramite un numero d'ordine che rappresenta l'indice (quelli appena descritti), è anche possibile definire **array associativi**, dove ogni elemento è individuato dalla coppia *chiave e valore*.

In pratica il numero d'ordine viene sostituito da una chiave d'identificazione che rappresenta la chiave d'accesso a ogni elemento. Con la seguente dichiarazione, per esempio, si definisce un array con tre elementi, in cui il primo è accessibile tramite la chiave "Luigi" e contiene il valore 18, il secondo è accessibile tramite la chiave "Marta" e contiene il valore 15, il terzo è accessibile tramite la chiave "Chiara" e contiene il valore 19:

```
$eta = array ("Luigi" => 18, "Marta" => 15, "Chiara" => 19)
```

Se si vuole stampare l'età corrispondente all'elemento con chiave "Marta", si usa l'istruzione:

```
echo "Marta ha $eta[Marta] anni"
```

che visualizza la riga:

```
Marta ha 15 anni
```

Per scorrere in sequenza un array associativo si usano le istruzioni **list()**, usata per assegnare valori a una lista di variabili con una sola operazione, e **each()**, che restituisce la coppia corrente chiave/valore di un array e incrementa il puntatore dell'array stesso.

Per esempio, con le istruzioni:

```
while (list($chiave, $valore) = each($eta)) {  
    echo "$chiave => $valore<br>"; }
```

si esegue un ciclo in cui ciascuna componente dell'array associativo \$eta viene riportata nelle variabili \$chiave e \$valore che sono visualizzate.

Come si può osservare nella trattazione appena presentata, in PHP le stringhe possono essere identificate in due modi equivalenti. Una stringa, infatti, può essere racchiusa tra apici singoli o doppi. Per esempio, le stringhe 'Marta' oppure "Marta" sono entrambe corrette.

Anche PHP mette a disposizione numerose funzioni per facilitare la realizzazione dei programmi. Nel seguito mostriamo le più utilizzate.

Funzioni

Elenchiamo le funzioni per lavorare con i numeri ([tabella 2](#)) e con le stringhe ([tabella 3](#)).

comando	esecuzione
Abs(\$x)	Restituisce il valore assoluto di x.
Pow(\$x,\$n)	Restituisce il valore di x elevato a n.
Sqrt(\$x)	Restituisce la radice quadrata di x.
Log(\$x)	Restituisce il logaritmo naturale dell'argomento.
Log10(\$x)	Restituisce il logaritmo in base 10 dell'argomento.
Round(\$x)	Arrotonda il numero decimale x al numero intero più vicino.
Rand()	Restituisce un numero casuale compreso tra 0 e 32767.
Srand	Inizializza il generatore di numeri casuali.

tabella 2 Lavorare con i numeri

comando	esecuzione
Strlen(\$Stringa)	Restituisce il numero di caratteri presenti nella stringa.
Strtolower(\$Stringa)	Trasforma la stringa in lettere minuscole.
Strtoupper(\$Stringa)	Trasforma la stringa in lettere maiuscole.
LTrim(\$Stringa)	Scarta gli spazi vuoti iniziali di una stringa.
RTrim(\$Stringa)	Scarta gli spazi vuoti finali di una stringa.
Trim(\$Stringa)	Scarta gli spazi vuoti iniziali e finali di una stringa.
Substr (\$Stringa, \$intero [, \$intero])	Restituisce una porzione della stringa: il secondo parametro indica l'inizio della porzione da estrarre, l'eventuale terzo parametro indica quanti caratteri sono estratti.
Strstr(\$Stringa, \$n)	Restituisce una sottocorrispondenza tratta da \$Stringa a partire da \$n.
strcmp (\$Stringa1,\$Stringa2)	Confronta \$Stringa1 con \$Stringa2.

tabella 3 Lavorare con le stringhe

Elenchiamo le funzioni per lavorare con date e orari ([tabella 4](#)) e i formati ammessi ([tabella 5](#)).

tabella 4 Date e orari

comando	esecuzione
Checkdate (\$mese, \$giorno, \$anno)	Verifica se i valori passati costituiscono una data valida.
Date("formato")	Restituisce la data corrente di sistema nel formato specificato.

tabella 5 Formato delle date

valore	descrizione	valore	descrizione
Y	anno su 4 cifre	y	anno su 2 cifre
m	mese numerico su 2 cifre (01-12)	n	mese numerico (1-12)
F	mese testuale ('January'- 'December')	M	mese testuale su 3 lettere ('Jan'-'Dec')
d	giorno del mese su due cifre (01-31)	j	giorno del mese (1-31)
w	giorno della settimana, numerico (0 = dom, 6 = sab)	D	giorno della settimana, su 3 lettere ('Sun'-'Sat')
H	ora su due cifre (00-23)	G	ora (0-23)
i	minuti su due cifre (00-59)	s	secondi su due cifre (00-59)



FISSA LE CONOSCENZE

1. Come viene individuato il codice PHP in una pagina HTML?
2. Qual è la sintassi dell'operatore ternario?
3. Spiega la differenza tra array associativi e array scalari.

2. HTML E PHP

Attraverso un **form HTML** il client è in grado di inviare dati al server. Nel form sono presenti elementi in cui l'utente inserisce dati (caselle di testo, pulsanti di opzione, caselle di riepilogo e caselle di controllo) e deve essere presente il tag:

```
<form method='post' action='elabora.php'>
```

Il tag indica la pagina che deve ricevere i dati (proprietà **Action**) e la modalità con cui li riceve (proprietà **Method**).

La proprietà Method può avere due valori: GET o POST.

■ **Modo GET:** consiste nell'accodare i dati all'indirizzo della pagina richiesta, facendo seguire il nome della pagina da un punto interrogativo e dalle coppie nome/valore dei dati che ci interessano.

Le diverse coppie sono separate dal segno &; quindi, immaginando di avere la pagina *elenco.php* che mostra i dati di uno studente, per visualizzare i dati di Mario Rossi dovremo richiamare la pagina in questo modo:

```
<a href='elenco.php?nome=Mario&cognome=Rossi'>
```

Quando la pagina *elenco.php* viene richiamata, mette a disposizione, al suo interno, l'array `$_GET`, dal quale si possono prelevare le variabili `$_GET[nome]` (con valore Mario) e `$_GET[cognome]` (con valore Rossi).

■ **Modo POST:** i dati vengono inviati, attraverso la richiesta HTTP che il browser invia al server, in maniera che questi non siano direttamente visibili per l'utente.

I dati passati attraverso il metodo POST sono memorizzati nell'array `$_POST`, che si comporta come l'array `$_GET` prima descritto. I valori sono a disposizione nelle variabili `$_POST['nome']` e `$_POST['cognome']`.

Poiché gli elementi nel form HTML sono caratterizzati da un nome (`<input type='text' name='codice'>`), nella pagina PHP è possibile prelevare i valori degli elementi del form usando una variabile PHP dove inserire i valori estratti con `$_GET` o `$_POST` a seconda della proprietà Method.

```
$CODICE = $_GET['codice']
```

Ora esaminiamo l'invio e il recupero dei dati, differenziando i casi in base al tipo di elemento presente nel form HTML.

Casella di testo

Se nella pagina HTML abbiamo la dichiarazione della seguente casella di testo:

```
<input type='text' name='NomeText'>
```

il dato contenuto nella casella di testo è reso disponibile nel modulo PHP nell'array `$_POST[NomeText]`, oppure `$_GET[NomeText]`, in base alle impostazioni di Method.

Esempio:

HTML	<form method='post' action='elabora.php'> <Cognome:input type='text' name='Cognome'>	Cognome: Rossi
PHP	<? \$Cognome = \$_POST[Cognome]; echo "Cognome = \$Cognome"; ?>	Cognome = Rossi

Casella di controllo (CheckBox)

Inserendo la dichiarazione di una CheckBox nel codice HTML:

```
<input type='checkbox' name='NomeCasella' value='valore' checked> etichetta
```

il dato contenuto nella CheckBox è reso disponibile nel modulo PHP nell'array `$_POST[NomeCasella]`, oppure `$_GET[NomeCasella]`. La presenza della proprietà `checked` comporta che la casella appaia sul video con il segno di spunta. In presenza di più caselle di controllo sul form il recupero dei dati avviene secondo le modalità descritte; perciò è consigliabile dare nomi diversi alle caselle.

HTML	<form method='post' action='CasellaControllo.php'> <input type='checkbox' name='Calcio' value='1' checked> Calcio <input type='checkbox' name='Nuoto' value='1'> Nuoto	<input checked="" type="checkbox"/> Calcio <input type="checkbox"/> Nuoto
PHP	<? \$Calcio = \$_POST[Calcio]; \$Nuoto = \$_POST[Nuoto]; If(\$Calcio == 1) echo 'Hai scelto Calcio '; If(\$Nuoto == 1) echo 'Hai scelto Nuoto'; ?>	Hai scelto Calcio

Caselle di riepilogo (ListBox)

Inserendo la dichiarazione di una ListBox nel codice HTML:

```
<select name='NomeListBox'>  
<option value='valore1'>etichetta1</option>  
<option value='valore2'>etichetta2</option>  
.....  
<option value='valoreN'>etichettaN</option>  
</select>
```

il valore selezionato è reso disponibile nel modulo PHP nell'array `$_POST[NomeListBox]`, oppure `$_GET[NomeListBox]`

HTML	<Form Method=Post Action="CasellaRiepilogo.php"> <Select name = "Sport"> <Option value = 1>Calcio</Option> <Option value = 2>Nuoto</Option> </Select>	<input checked="" type="radio"/> Calcio <input type="radio"/> Nuoto
PHP	<? \$Sport = \$_POST[Sport]; If(\$Sport == 1) echo "Hai scelto Calcio"; Else echo "Hai scelto Nuoto"; ?>	Hai scelto Calcio

3. PASSAGGIO DI PARAMETRI IN PHP

Quando si realizza un sito web formato da pagine diverse, è importante poter passare le informazioni tra una pagina e l'altra o conservare informazioni reperibili da pagine differenti. Nella Lezione precedente abbiamo già visto come passare i dati da un form HTML a una pagina PHP (metodi POST e GET).

Vediamo ora altri metodi per mantenere le informazioni tra le varie pagine navigate: si possono usare i **cookie** o le **sессии**. Le differenze tra i due strumenti sono numerose. La principale sta nel fatto che, mentre la sessione si esaurisce con il termine della singola navigazione (per esempio, l'utente chiude il browser), il cookie ha una durata superiore, che consente di mantenere informazioni anche per le sessioni successive. Per questo, capita spesso che sia richiesta all'utente l'autorizzazione a utilizzare i cookie.

■ I cookie

I cookie sono frammenti di testo inviati dal server al client.

```
setcookie('nome_cookie', 'valore', scadenza)
```

L'istruzione permette di impostare un cookie. In essa:

- il primo parametro specifica il nome identificativo del cookie;
- il secondo parametro specifica il valore del cookie;
- il terzo parametro imposta la scadenza in secondi: se non si imposta una data di scadenza, il cookie non scadrà mai.

Con l'istruzione:

```
setcookie('nome_utente', 'Mario', time() +3600)
```

viene **creato un cookie** denominato `nome_utente`, al quale è assegnato il valore 'Mario'. Esso scadrà tra un'ora, in quanto `time()` indica l'ora corrente. Il valore del cookie è prelevato nel modulo PHP con la seguente istruzione:

```
$nome = $_COOKIE['nome_utente']
```

Se si vuole **cancellare il cookie**, basta richiamarlo senza specificare nessun valore:

```
setcookie('nome_utente')
```

La funzione "setcookie" ha una particolarità: può essere utilizzata solo nella parte iniziale di uno script; più precisamente, essa può essere richiamata esclusivamente prima che una qualsiasi porzione di pagina sia inviata al client (tipicamente, prima del tag <HTML>).

■ Le variabili di sessione

La sessione, così come i cookie, permette di mantenere informazioni tra le varie pagine navigate. È un file di testo che contiene diverse informazioni sull'utente. Per dare il via a una sessione si utilizza la funzione:

```
session_start()
```

Per registrare **variabili di sessione**, bisogna inserirle nell'array associativo globale `$_SESSION` nel seguente modo:

```
$_SESSION['nome_var'] = 'valore'
```

Da questo momento in poi l'array `$_SESSION` conterrà il valore associato all'indice `nome_var`, il che corrisponde all'aver salvato, all'interno della sessione, la variabile `nome_var` a cui è associato il valore. Se vogliamo salvare il nome e il cognome di un utente nella variabile `$_SESSION`, scriviamo il seguente codice:

```
<?      $_SESSION['nome'] = 'Mario'  
        $_SESSION['cognome'] = 'Rossi';  
?>
```

In seguito, per **prelevare** il nome e il cognome dell'utente dalla variabile di sessione, scriveremo queste istruzioni:

```
<?      $nome = $_SESSION['nome'] ;  
        $cognome = $_SESSION['cognome'];  
?>
```

Per cancellare una variabile di sessione si usa:

```
unset ($nome_var)
```

Infine, per chiudere completamente una sessione e quindi eliminare il file sul server contenente i dati di sessione si usa la funzione:

```
session_destroy()
```

ESEMPIO

Si vogliono contare il numero di accessi che si fanno a una pagina.

Per contarli è possibile usare una variabile di sessione nel modo seguente:

```
<? //conteggio degli accessi a una pagina  
    session_start();  
    if(!isset($_SESSION['count']))  
        $_SESSION['count'] = 0;  
    else  
        $_SESSION['count']++;  
    echo "hai visitato questa pagina $_SESSION[count] volte";  
?>
```

La funzione `isset` restituisce VERO se la variabile COUNT è stata settata come variabile di sessione, FALSO altrimenti.

LABORATORIO

IL PROBLEMA Costruire una pagina web la quale, dopo aver richiesto nome, cognome e sesso di un utente, richiami una seconda pagina in cui vengono visualizzati un messaggio di saluto per l'utente inserito, un menu a tendina con vari sport e un pulsante che richiama una terza pagina, dove vengono visualizzati nome e cognome dell'utente e il suo sport preferito.

L'ANALISI

Scriviamo una pagina HTML con due caselle di testo per inserire i dati e due buttoni: uno (Annulla) per annullare e l'altro (Visualizza) che richiama una pagina PHP. Questa riceve i due valori con il metodo POST, mostra un menu a tendina per la scelta dello sport e il bottone Continua, con cui richiama un'altra pagina PHP, passando i valori ricevuti dall'HTML, con il metodo SESSION.

LA CODIFICA

visualizza.html

```
<html>
<head></head>
<body>
<form method='post' action='visualizza.php'>
Cognome: <input type='text' name='cognome' size='10'>
<p>&nbsp;</p>
Nome: <input type='text' name='nome' size='10'>
<p>&nbsp;</p>
Sesso: <input type='radio' name='sesso' value='1'> Maschio
<input type='radio' name='sesso' value='2'> Femmina
<p>&nbsp;</p>
<input type='submit' value='Visualizza' > &nbsp; &nbsp; &nbsp;
<input type='reset' value='Annulla'>
</form>
</body>
</html>
```

Dalla pagina *visualizza.html* vengono passati i parametri alla pagina *visualizza.php* con il metodo POST.

visualizza.php

```
<?php
    //recupera i dati passati dal form html
    $cognome = $_POST['cognome'];
    $nome = $_POST['nome'];
    $sesso = $_POST['sesso'];
    if ($sesso == 1)
        print "benvenuto $cognome $nome";
    else
        print "benvenuta $cognome $nome";
    // prepara i dati da passare alla seconda pagina
    $_SESSION['cognome']= $cognome;
    $_SESSION['nome'] = $nome;
    $_SESSION['sesso']= $sesso;
    print "<form method='post' action='visualizzaSession.php'>";
```

```

print "<br><br>Quale sport preferisci?<br><br> <select name='sport'>";
print "<option value='1'> calcio </option>";
print "<option value='2'> tennis </option>";
print "<option value='3'> nuoto </option>";
print "<option value='4'> ciclismo </option>&nbsp; &nbsp;";
print "<input type='submit' value='Continua'>";
print "</form>";
?>

```

La terza pagina (*visualizzaSession.php*) è richiamata, passando i parametri con il metodo SESSION.

visualizzaSession.php

```

<?php
//recupera i dati passati nella variabile sessione
$cognome = $_SESSION['cognome'];
$nome = $_SESSION['nome'];
$sesso = $_SESSION['sesso'];
$sport = $_POST['sport'];
if ($sesso == 1)
print "Signor $cognome $nome ";
else
print "Signora $cognome $nome";
if ($sport == 1)
print "<br> il tuo sport preferito è il calcio";
if ($sport == 2)
print "<br> il tuo sport preferito è il tennis";
if ($sport == 3)
print "<br> il tuo sport preferito è il nuoto";
if ($sport == 4)
print "<br> il tuo sport preferito è il ciclismo";
?>

```

IL RISULTATO

The figure consists of three screenshots illustrating the interaction between a form and a session-based script:

- Screenshot 1:** A browser window titled "localhost/U2L4/visualizza.html". It contains a form with fields for "Cognome" (Marrone), "Nome" (Velia), and "Sesso" (radio buttons for "Maschio" and "Femmina"). Below the form are two buttons: "Visualizza" and "Annulla".
- Screenshot 2:** A browser window titled "localhost/U2L4/visualizza.php". It displays the message "benvenuta Marrone Velia" followed by the question "Quale sport preferisci?". Below the question is a dropdown menu containing four options: "calcio", "tennis", "nuoto", and "ciclismo". The "calcio" option is highlighted with a dark gray background.
- Screenshot 3:** A browser window titled "localhost/U2L4/visualizzaSession.php". It shows the output "Signora Marrone Velia" followed by the message "il tuo sport preferito e' il tennis".

FISSA LE CONOSCENZE

1. Qual è la differenza tra POST e GET?
2. Come si crea e si distrugge un cookie?
3. Che cosa si intende per sessione?

4. CONNESSIONE AL DB E VISUALIZZAZIONE DATI

Nelle lezioni che seguono si farà uso del **MySQLi** al posto del driver classico MySQL. MySQLi è la nuova interfaccia tra PHP e un database MySQL e può essere usata da qualsiasi versione di MySQL superiore alla 4.1.3. Rispetto al MySQL, questa estensione è completamente Object Oriented, implementa nuove caratteristiche in modo standard e facilita il debug per la ricerca di problematiche. Inoltre, ricordiamo che, dalla versione di PHP 7.0, il driver MySQL non è più presente. Le pagine HTML per brevità sono costruite con i costrutti elementari e senza far ricorso ai CSS.

Per realizzare il collegamento tra una pagina PHP e un database MySQL, è necessario innanzitutto aprire una **connessione con il database**.

A tale scopo si utilizza la funzione `mySQLi_connect`:

```
mySQLi_connect(server, utente, password, nome database);
```

alla quale viene passato l'indirizzo in cui si trova il server MySQL (*localhost*, quando opera sulla stessa macchina del Web Server), il nome utente con il quale ci si collega, la password e il nome del database che si vuole utilizzare. Spesso si utilizza root come utente e la password vuota:

```
$conn = mySQLi_connect ("localhost", "root", " ",$dbname);
```

Se il collegamento va a buon fine, la funzione restituisce un identificativo alla connessione stessa, altrimenti restituisce il valore False.

Al termine delle operazioni è necessario **chiudere la connessione** con l'`istruzione`:

```
mySQLi_close($conn);
```

LABORATORIO

IL PROBLEMA Creare uno script per aprire una connessione a un database, controllando che non ci siano errori.

LA CODIFICA

`connessione.php`

```
<html><head></head>
<body>
<p align="center"><strong>Connessione al database</strong><br><br><br>
<?php
$hostname = "localhost";
$username = "root";
$password = "";
$dbname = "dblogin";
//connessione al server sql con selezione del database
$conn = mysqli_connect($hostname, $username, $password, $dbname);
if(!$conn)
```

```
die ("<br><br><strong>Errore nella connessione</strong>");  
else  
    "<br><br><strong>Connessione avvenuta correttamente</strong><br><br>";  
mysqli_close($conn);  
print "<strong>Ora la connessione è chiusa</strong>";  
?>  
</body>  
</html>
```

IL RISULTATO



Vediamo ora come eseguire una query e visualizzare i risultati, dopo aver effettuato una connessione al database.

La funzione `mysqli_query()` permette di inviare una query al server. Tale funzione restituisce `False` se ci sono errori, mentre, se l'operazione è andata a buon fine, restituisce l'indirizzo del buffer in cui ha inserito i risultati della query. Il seguente codice:

```
$Risultato = mysqli_query($query);  
if(!$Risultato)  
    print 'errore nel comando';
```

permette di verificare se la query contenuta nella variabile `$query` è andata a buon fine; se è così in `$Risultato` è contenuto il puntatore al buffer che contiene i risultati, altrimenti assume il valore `False`.

Per estrarre i dati dal buffer è necessario fare uso della funzione: `mysqli_fetch_array()`, che preleva e restituisce un record dei risultati dal buffer. Se il buffer è vuoto, restituisce il valore `False`.

LO SAI CHE

Buffer

Area di memoria dove vengono tenuti dei dati in modo provvisorio.

LABORATORIO

IL PROBLEMA Visualizzare i dati del prodotto presente in magazzino corrispondente al codice inserito in input.

L'ANALISI Il codice del prodotto desiderato viene preso in input tramite il modulo HTML (`codiceProdotto.html`). Quando si seleziona il pulsante Visualizza, viene eseguita la pagina `codice.php`, dove è impostata una query SQL sulla tabella Prodotti con la clausola `id_prodotto = codice`. Se la query restituisce un insieme vuoto, vuol dire che non esiste il codice inserito; in caso contrario vengono visualizzati i dati del prodotto. Il pulsante Annulla inserito nella pagina HTML viene usato per pulire la casella di testo.

LA CODIFICA

codiceProdotto.html

```
<html>
<head></head>
<body>
<p><font size='5'><strong>Ricerca prodotto</strong></font><br><br><br>
<form method='post' action='codice.php'>
Codice prodotto: <input type='text' name='codice' size='10'>
<p>&nbsp;</p>
<input type='submit' value='Visualizza' &nbsp; &nbsp; &nbsp; ;
<input type='reset' value='Annulla'>
</form>
</body>
</html>
```

codice.php

```
<?php
$hostname = "localhost";
$username = "root";
$password = "";
$dbname = "magazzino2";
//connessione al server SQL
$conn = mysqli_connect($hostname, $username, $password, $dbname);
if (!$conn)
{
    die ("errore nella connessione");
    exit();
}
//recupera i dati passati dal form html
$codice = $_POST['codice'];
$query = "Select * from prodotti where id_prodotto = '$codice'";
$risultato= mysqli_query($conn,$query);
if (!$risultato)
{
    print "errore nel comando";
    exit();
}
$riga = mysqli_fetch_array($risultato);
if ($riga)
{
    print "descrizione: ".$riga['descrizione']."<br>";
    print "prezzo unitario: ".$riga['prezzo']."<br>";
    print "quantita:".$riga['quantita'];
}
else
{
    print "Attenzione!!! codice prodotto $codice non presente";
}
mysqli_close($conn);
?>
```

IL RISULTATO

The screenshot shows a search interface with the URL `localhost/u215/codiceProdotto.html`. The search term `pr03` was entered. The results are displayed in a box on the right:

```
descrizione: gomma  
prezzo unitario: 2  
quantita':100
```

Quando l'esecuzione di una query fornisce un insieme di record, per visualizzare tutti i record selezionati bisogna usare la funzione `mysqli_fetch_array()` all'interno di un ciclo, che termina quando la funzione restituisce il valore False:

```
$riga = mysqli_fetch_array($risultato);
while($riga)
{
    // vengono scritti i valori dei campi del record corrente
    print "Codice: $riga['id_prodotto'] <br>";
    print "Descrizione: $riga['descrizione'] <br>";
    print "Prezzo: $riga['prezzo']<br>";
    print "Quantità: $riga['quantita']<br>";
    //si sposta sulla riga successiva
    $riga = mysqli_fetch_array($risultato);
}
```

Se invece si vuole sapere quante righe sono state restituite dall'esecuzione della query, si può utilizzare la funzione:

```
mysqli_num_rows($risultato);
```

che restituisce il **numero di righe** presenti nel buffer.

LABORATORIO

IL PROBLEMA Visualizzare i dati dei prodotti presenti in magazzino in ordine crescente di prezzo.

L'ANALISI Una volta stabilita la connessione con il database Magazzino, si manda in esecuzione il comando SQL, che estrae i dati dalla tabella Prodotti ordinati per prezzo (clausola Order by prezzo). Attraverso un ciclo di scansione sul risultato della query si visualizzano i dati prelevati dal database all'interno di una tabella.

LA CODIFICA

elencoProdotti.php

```
<html>
<head></head>
<body><p><font size='5'><strong>Elenco prodotti</strong><br><br><br><?php
```

```

$hostname = "localhost";
$username = "root";
$password = "";
$dbname = "magazzino2";
//connessione al server SQL
$conn = mysqli_connect($hostname, $username, $password, $dbname);
if (!$conn)
{
    print "errore nella connessione";
    exit();
}
$query = "Select * from prodotti order by prezzo";
$risultato = mysqli_query($conn,$query);
if (!$risultato)
{
    print "errore nel comando";
    exit();
}
$riga = mysqli_fetch_array($risultato);
if ($riga)
{
    print "<table border='1'>";
    print "<tr><td> CODICE </td><td> DESCRIZIONE </td>";
    print "<td> PREZZO </td><td> QUANTITÀ</td></tr>";
    while($riga){
        print "<tr><td>".$riga['id_prodotto']."</td>";
        print "<td>".$riga['descrizione']."</td>";
        print "<td>".$riga['prezzo']."</td>";
        print "<td>".$riga['quantita']."</td></tr>";
        $riga = mysqli_fetch_array($risultato);
    }
    print "</table>";
}
else
    print "Attenzione!!! non sono presenti prodotti".
mysqli_close($conn);
?>
</body>
</html>

```

IL RISULTATO

CODICE	DESCRIZIONE	PREZZO	QUANTITA'
pro2	matita	1	80
pro3	gomma	2	100
pro4	penna	2	150
pro1	quaderno	2.5	200
pr70	notes	3.5	70

5. INSERIMENTO E MODIFICA DATI

Per inserire, modificare o eliminare dati in una tabella attraverso uno script PHP, è necessario lanciare l'esecuzione di una query di aggiornamento attraverso il richiamo alla funzione:

```
mysqli_real_query($connessione,$query);
```

che riceve una stringa contenente la query e la variabile di connessione e restituisce il valore True se l'operazione va a buon fine, False se l'operazione non viene eseguita.

La stringa \$query dovrà contenere il comando SQL che si vuole eseguire. Per esempio, se si vuole inserire una nuova riga all'interno di una tabella, si usa il comando SQL INSERT INTO che ha la seguente sintassi:

```
INSERT INTO nome_tabella VALUES (valore1,valore2, ...)
```

oppure:

```
INSERT INTO nome_tabella (nome_campo1, nome_campo2, ...) VALUES (valore1,valore2, ...)
```

Analogamente al comando INSERT INTO, si possono utilizzare i comandi DELETE e UPDATE secondo la sintassi SQL.

Dopo l'esecuzione di una query di aggiornamento (INSERT, UPDATE, DELETE) è possibile richiamare la funzione:

```
Mysqli_affected_rows($connessione)
```

che restituisce il numero delle righe della tabella inserite (o modificate o cancellate) dal comando eseguito.

LABORATORIO

IL PROBLEMA Inserire i dati di un nuovo prodotto introdotto in magazzino.

L'ANALISI I dati del nuovo prodotto vengono immessi attraverso un modulo HTML, che conterrà il pulsante Inserisci attraverso il quale viene richiamata la pagina *insProdotto.php*. In essa metteremo il comando SQL, che permette di inserire i dati del prodotto nel database. Se l'esecuzione della query genera errore, l'errore viene segnalato, altrimenti i dati sono inseriti nella tabella. Nella pagina inseriamo anche il pulsante Reset per pulire le caselle di testo.

LA CODIFICA

insProdotto.htm

```
<html>
<head></head>
<body>
<form method='post' action='insProdotto.php'>
Codice: <input type='text' name='codice' size='10'> <br><br>
```

```

Descrizione: <input type='text' name='descrizione' size='40'> <br><br>
Prezzo: <input type='text' name='prezzo' size='40'> <br><br>
Quantità: <input type='text' name='quantita' size='40'> <br><br>
<input type='submit' value='Inserisci'> &nbsp; &nbsp; &nbsp;
<input type='reset' value='Reset'>
</form>
</body>
</html>

```

insProdotto.php

```

<?php
    $hostname = "localhost";
    $username = "root";
    $password = "";
    $dbname = "magazzino2";
    //connessione al server SQL
    $conn = mysqli_connect($hostname, $username, $password,$dbname);
    if (!$conn) {
        die ("errore nella connessione");
        exit();
    }
    //recupera i dati passati dal form html
    $codice = $_POST['codice'];
    $descrizione = $_POST['descrizione'];
    $prezzo = $_POST['prezzo'];
    $quantità = $_POST['quantita'];
    $query = "INSERT INTO prodotti (id_prodotto, descrizione, prezzo, quantita) VALUES
    ('$codice', '$descrizione', $prezzo, $quantità')";
    $risultato = mysqli_real_query($conn,$query);
    $numRighe = mysqli_affected_rows($conn);
    if (!$risultato)
        print "errore nell'inserimento";
    else
        print "registrazione avvenuta correttamente<br> sono state inserite $numRighe righe";
    mysqli_close($conn);
?>

```

IL RISULTATO

The image shows two side-by-side browser windows. Both windows have a header bar with back, forward, and refresh buttons, and a URL field set to "localhost/u2l6/insProdotto.php".

Left Window (Form View):

- URL: localhost/u2l6/insProdotto.html
- Form fields:
 - Codice: pr10
 - Descrizione: notes
 - Prezzo : 3.5
 - Quantità : 70
- Buttons: Inserisci, Reset

Right Window (Result View):

- URL: localhost/u2l6/InsProdotto.php
- Content area displays the message: "registrazione avvenuta correttamente" followed by "sono state inserite le righe"

6. LOGIN

Vediamo ora come gestire l'accesso a un sito qualunque a cui sono ammessi solo gli utenti registrati.

LABORATORIO

IL PROBLEMA Gestire l'accesso a un sito tramite l'inserimento di username e password.

L'ANALISI Per realizzare questa applicazione, usiamo un database in cui è presente una tabella contenente le credenziali di tutti gli utenti registrati. La nostra applicazione presenterà una pagina iniziale, in cui l'utente ha la possibilità di effettuare l'accesso, se è già registrato, o di registrarsi, se non possiede le credenziali. Nel caso in cui le credenziali inserite non siano presenti nel database, è prevista la possibilità di reinserire i dati corretti o scegliere di registrarsi.

LA CODIFICA

login.html

```
<html>
<head></head>
<body>
<br><br>
<form method='post' action='inserisciLogin.php'>
<input type='submit' value='Accedi'>&nbsp;&nbsp;&nbsp;
</form>
<br><br>
<form method='post' action='inserimento.php'>
<input type='submit' value='Registrati'>
</form>
</body>
</html>
```

inserisciLogin.php

```
<?php
print "<form method='post' action='login.php'>";
print "Username: <input type='text' name='user' size='10'><br><br>";
print "Password: <input type='text' name='psw' size='10'><br><br>";
print "<input type='submit' value='Login'>&nbsp;&nbsp;&nbsp;";
print "<input type='reset' value='Reset'>";
print "</form>";
?>
```

inserimento.php

```
<?php
print "<p><form method='post' action='registrati.php'>";
print "Cognome: <input type='text' name='cognome' size='10'><br><br>";
print "Nome: <input type='text' name='nome' size='10'><br><br>";
```

```
print "Username: <input type='text' name='user' size='10'><br><br>";
print "Password: <input type='text' name='psw' size='10'>";
print "<br><br><br><input type='reset' value='Reset'>&nbsp; &nbsp; &nbsp;";
print "<input type='submit' value='Registrati'>";
print "</form>";
?>
```

login.php

```
<?php
$hostname = "localhost";
$username = "root";
$password = "";
$dbname = "dblogin";
//connessione al server SQL
$conn = mysqli_connect($hostname, $username, $password, $dbname);
if (!$conn)
{
    print "errore nella connessione";
    exit();
}
//recupera i dati passati dal form html
$user = $_POST['user'];
$psw = $_POST['psw'];
if ($user == "" || $psw == "")
{
    print "campi vuoti";
    print "<br><a href='inseriscilogin.php'>torna al login</a>";
}
else
{
    $query = "Select * from login where user = '$user' && psw = '$psw'";
    $risultato = mysqli_query($conn, $query);
    if (!$risultato)
    {
        print "errore nel comando";
        exit();
    }
    $riga = mysqli_fetch_array($risultato);
    if ($riga)
        print "Benvenuto". $riga['nome']. ".$riga['cognome']";
    else
    {
        print "username o password errate";
        print "<br><a href='inseriscilogin.php'>torna al login</a>";
        print "<br><a href='inserimento.php'>registrati</a>";
    }
}
mysqli_close($conn);
?>
```

registrati.php

```
<?php
    $hostname = "localhost";
    $username = "root";
    $password = "";
    $dbname = "dblogin";
    //connessione al server SQL
    $conn = mysqli_connect($hostname, $username, $password, $dbname);
    if (!$conn) {
        print "errore nella connessione";
        exit();
    }
    //recupera i dati passati dal form html
    $user = $_POST['user'];
    $psw = $_POST['psw'];
    $nome = $_POST['nome'];
    $cognome = $_POST['cognome'];
    $query = "INSERT INTO login ('cognome', 'nome', 'user', 'psw')
              VALUES ('$cognome', '$nome', '$user', '$psw')";
    $risultato = mysqli_real_query($conn,$query);
    if (!$risultato)
        print "errore nell'inserimento: username già presente";
    else
        print "registrazione avvenuta correttamente";
    mysqli_close($conn);
?>
```

IL RISULTATO

The figure consists of four screenshots of web browser windows. The first screenshot shows a login page with two buttons: 'Accedi' (Login) and 'Registrati' (Register). The second screenshot shows a registration page where the 'Username' field contains 'Iacobelli' and the 'Password' field contains five asterisks. It also features 'Login' and 'Reset' buttons. The third screenshot shows a confirmation page with the message 'Benvenuto Cesare Iacobelli'. The fourth screenshot shows an insertion page with four input fields labeled 'Cognome', 'Nome', 'Username', and 'Password', each preceded by a label. Below these fields are 'Reset' and 'Registrati' buttons.

7. IMPORTARE ED ESPORTARE DATI

Quando si deve popolare un database aziendale, la tecnica illustrata nelle precedenti lezioni può risultare poco efficiente, poiché prevede di inserire da tastiera i dati per ciascun record. Se la cardinalità della tabella è elevata, questa operazione può richiedere molto tempo e aumentare il rischio di commettere errori di digitazione.

Nella realtà possono presentarsi diversi scenari e per ciascuno si può adottare la soluzione adeguata. Per esempio, può verificarsi il caso in cui i dati siano memorizzati in un foglio Excel o siano contenuti in un file di testo.

LO SAI CHE

Cardinalità

Numero di righe (o n-ple) presenti in una tabella.

■ Importa dati usando phpMyAdmin

Utilizzando phpMyAdmin, l'importazione/esportazione dei dati è semplice grazie ai tasti presenti sulla barra dei comandi (fig. 1).

Tramite phpMyAdmin si possono importare/esportare interi database o singole tabelle.

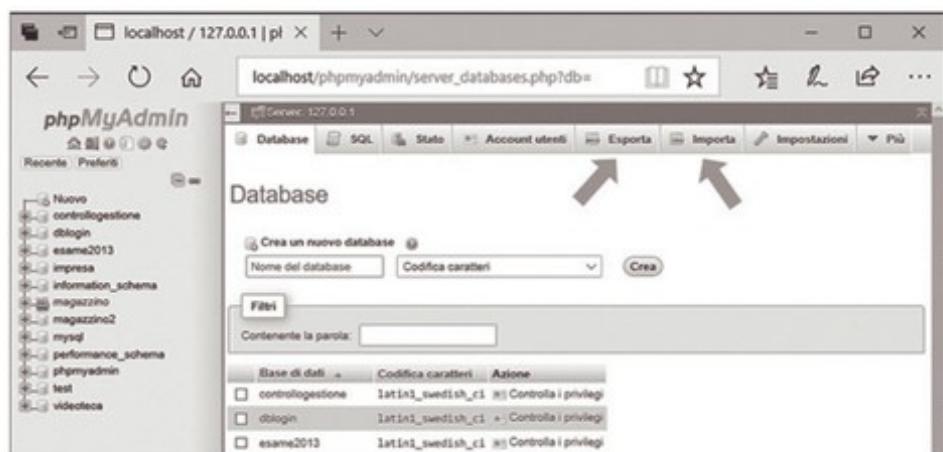


fig. 1 Importa/esporta dati.

È possibile scegliere il file da cui prelevare i dati e il suo formato. Cliccando su Esegui, i dati vengono importati nel database (fig. 2). Se per esempio vogliamo importare dei dati da un foglio Excel, possiamo salvare il file in formato XML e importarlo.



fig. 2 Scelta del file da importare.

Lo stesso discorso vale per l'esportazione: viene presentata la videata della **fig. 3** dando la possibilità di scegliere il formato del file da esportare. Cliccando sul tasto Esegui, si può scegliere se salvare o aprire il file generato (**fig. 4**).

fig. 3 Scegliere il formato del file da esportare.

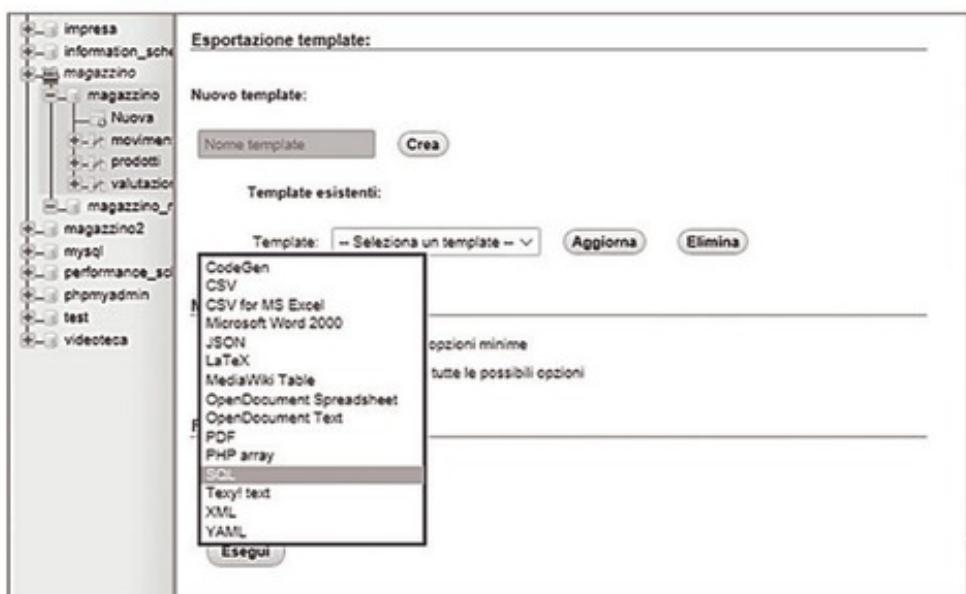


fig. 4 Salvare o aprire il file esportato.



■ Importare dati tramite programmi

È possibile popolare un database, scrivendo un apposito programma, come è indicato nell'esercizio guidato che segue.

LABORATORIO

IL PROBLEMA Inserisci nella base di dati le informazioni relative ai nuovi prodotti contenute in un file di testo.

L'ANALISI Dopo aver effettuato la connessione con la base di dati, viene eseguito un ciclo di lettura su ogni riga del file di testo: il ciclo legge una riga del file e per ogni riga si separano i campi con il **metodo explode**, che restituisce i singoli dati in un vettore.

Il ciclo viene reiterato per ogni riga fino alla fine del file.

Può essere così preparato il comando SQL INSERT per inserire il nuovo prodotto nella tabella.

LA CODIFICA

```
<?php
    $hostname = "localhost";
    $username = "root";
    $password = "";
    $dbname = "magazzino2";
    //connessione al server SQL
    $conn = mysqli_connect($hostname, $username, $password, $dbname);
    if(!$conn){
        print "errore nella connessione";
        exit();
    }
    // apertura del file contenente i dati da inserire
    if(!$fp = fopen("prodotti.txt", "r")){
    {
        print "errore nell'apertura del file";
        exit();
    }
    // si prepara la prima parte della query
    $query = "INSERT INTO prodotti (id_prodotto, descrizione, prezzo, quantita) VALUES ";
    //per ogni riga del file
    while (!feof($fp))
    {
        /*lettura di una riga del file */
        $line = fgets($fp);
        //recupero dei dati
        $dati = explode(' ', $line);
        $codice = $dati[0];
        $prezzo = $dati[1];
        $quantità = $dati[2];
        $descrizione = $dati[3];
        // aggiunta dei dati alla stringa della query
        $query .= "('$codice', '$descrizione', '$prezzo', '$quantità'),";
    }
    /*chiusura del file*/
    fclose($fp);
    //completamento della query: togliamo la , e inseriamo il ; finale
    $query = substr($query, 0, strlen($query)-1);
    $query .= ";";
    // esecuzione della insert
    $risultato = mysqli_real_query($conn, $query);
    $numRighe = mysqli_affected_rows($conn);
    if (!$risultato)
        print "errore nell'inserimento";
    else
        print "sono state inserite ".$numRighe. "<br>";
    mysqli_close($conn);
?>
```

Si lascia allo studente la definizione del programma che realizza l'esportazione dei dati dalla tabella Prodotti a un file di testo.

8. ESERCIZIO COMPLETO IN PHP

In questa Lezione prendiamo in esame in tutte le sue parti un esercizio complesso da svolgere in PHP.

LABORATORIO

IL PROBLEMA Si vuole realizzare un sito web per la gestione online di un museo, in particolare si vogliono implementare le seguenti richieste:

1. indicare il numero di biglietti emessi per una determinata esposizione;
2. calcolare il ricavato della vendita dei biglietti di una data esposizione.

Il database di riferimento è il seguente:

Visita (id_visita, titolo, tipo, tariffa, dataInizio, dataFine);

Biglietto (id_biglietto, dataValidità, cod_visita, cod_categoria);

Servizio (id_servizio, descrizione, prezzo);

Categoria (id_categoria, descrizione, tipoDocumento, percentualeSconto);

Associare (id_biglietto, id_servizio).

L'ANALISI Dobbiamo progettare il layout del sito per la gestione del museo e quindi realizzare i programmi che si interfaceranno con il database.

La prima operazione da fare è disegnare il layout delle pagine che compongono il sito web.

Nella Home page HTML (*esposizione.html*), saranno presenti: una casella di testo in cui inserire il codice dell'esposizione di cui cercare le informazioni; un tasto CERCA per richiamare la pagina successiva, che conterrà i dati relativi all'esposizione; i risultati delle query necessarie per indicare il numero di biglietti emessi e il ricavato. Il tasto CERCA richiama il programma *esposizione.php*.

L'INTERFACCIA

MUSEO ONLINE Codice esposizione <input type="text"/> <input type="button" value="CERCA"/>	ESPOSIZIONE Codice <input type="text"/> Titolo <input type="text"/> N. Biglietti <input type="text"/> Ricavo totale <input type="text"/>
--	---

LA CODIFICA PHP

esposizione.html

```
<html>
<head> <title> Esposizione </title>
</head><body>
<h4 align='center'> MUSEO ONLINE </h4>
<form name='invia' method='post' action='esposizione.php'>
<br>Codice esposizione: <input type='text' name='codice_esposizione' size='10'> <br>
```

```
<input type='submit' value='CERCA' >
</form> </body> </html>
```

esposizione.php

```
<html><head><title>museo </title></head>
<body>
<p align="center"><font size="3"><strong><u> ESPOSIZIONE </u></strong></font></p>
<?php
    $codice = $_POST['codice_esposizione'];
    print "<center> Codice = $codice </center>";
    // Apre la connessione con il data base
    $hostname = "localhost";
    $username = "root";
    $password = "";
    $dbname = "museo";
    //connessione al server SQL
    $conn = mysqli_connect($hostname, $username, $password, $dbname);
    if(!$conn){
        print "errore nella connessione";
        exit();}
    $query = "Select id_visita, titolo, tariffa
        From Visita      Where id_Visita = '$codice'";
    //recupera i dati
    $risultato = mysqli_query($conn,$query);
    $riga = mysqli_fetch_array($risultato);
    // Se $riga = false non ci sono righe di risultato
    if (!$riga) {
        echo "<font size=5><u>Attenzione l'esposizione richiesta non esiste </u></font>";
        exit;}
    $id_visita = $riga['id_visita'];
    $titolo = $riga['titolo'];
    $tariffa = $riga['tariffa'];
    print "<center> Titolo = $titolo </center>";
    //conta il numero di biglietti
    $query = "Select Count(*) As numeroBiglietti From Biglietto
                Where cod_visita = '$codice'";
    $risultato = mysqli_query($conn,$query);
    //recupera i dati
    $riga = mysqli_fetch_array($risultato);
    print "N. biglietti: " . $riga ['numeroBiglietti'];
    //calcola il ricavo
    $query = "SELECT Sum(tariffa - (tariffa * percentualeSconto / 100)) as totale
                from Visita,Biglietto,Categoria
                where id_visita = cod_visita and cod_categoria = id_categoria
                and cod_visita = '$codice'";
    $risultato = mysqli_query($conn,$query);
    $riga = mysqli_fetch_array($risultato);
    print "<br>Ricavo totale: " . $riga ['totale'];
    mysqli_close($conn);
?>
</body></html>
```



RIPASSIAMO INSIEME

Linguaggio PHP

PHP è un prodotto **open source** e funziona su moltissime piattaforme. Nella Lezione sono illustrate le **variabili**, gli **operatori**, le **strutture** e le **funzioni** principali.

HTML e PHP

Attraverso un **form HTML** il client è in grado di inviare dei dati al server; nel form vi sono elementi in cui l'utente inserisce dati.

Nella Lezione si esamina l'**invio** e il **recupero di dati** a seconda del tipo di elemento presente nel form HTML.

Passaggio di parametri in PHP

Le informazioni presenti in più pagine dinamiche possono essere salvate nei **cookie**, oppure nelle **variabili di sessione**. I cookie sono piccoli file memorizzati sul client per un tempo stabilito dall'utente; le variabili di sessione, invece, si esauriscono con il termine della singola navigazione.

Connessione al database e visualizzazione dati

Per realizzare il collegamento tra una pagina PHP e un database MySQL è necessario prima aprire una **connessione**; poi si possono eseguire **query** per recuperare

i dati memorizzati nelle tabelle e visualizzare i **risultati**.

Se l'esecuzione della query è andata a buon fine, viene restituito l'indirizzo del **buffer** in cui sono stati inseriti i risultati della query.

Inserimento e modifica dati

Per inserire, modificare o eliminare i dati in una tabella attraverso uno script PHP, è necessario lanciare l'esecuzione di una **query di aggiornamento**.

Login

In questa Lezione viene illustrato come realizzare la **verifica** di un accesso tramite username e password e l'eventuale inserimento nel database di un **nuovo utente**.

Importare ed esportare dati

In questa Lezione viene descritto come **creare**, **modificare** e **cancellare** un database in ambiente phpMyAdmin e tramite la scrittura di query SQL.

Esercizio completo

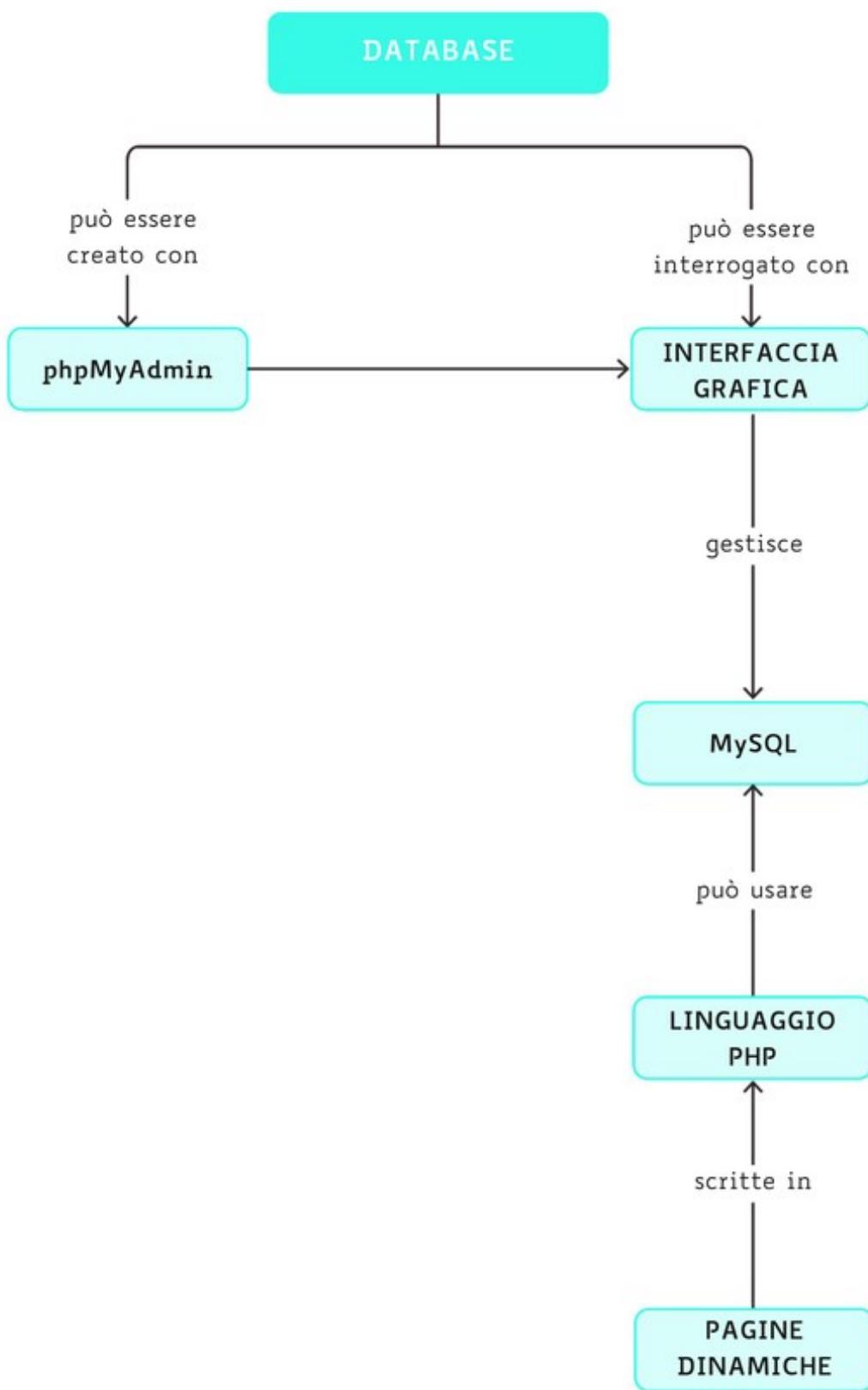
In questa Lezione viene presentato un esercizio svolto in tutte le sue parti con l'uso del linguaggio PHP.



MAPPA CONCETTUALE



MAPPA
MODIFICABILE



TEST**TEST**

Svolgi il test interattivo

Vero o falso?

1. Un programma PHP è eseguito dal browser. V F
2. Uno script PHP deve essere inserito nella sezione <head> di una pagina HTML. V F
3. La variabile \$_POST è un array. V F
4. Il valore inserito in una casella di testo è disponibile in \$_POST. V F
5. Una variabile di sessione non ha scadenza. V F
6. Per aprire una connessione con il server MySQL si usa la funzione mysqli_select_db. V F
7. Per eseguire una query si usa la funzione mysqli_query. V F

Scelta multipla (una sola è la risposta esatta)

8. Quale di queste è una variabile PHP?
 A codice C &codice
 B \$codice D Codice
 9. Quale delle seguenti istruzioni permette di acquisire un cognome da un form HTML?
 A \$Cognome = \$_POST['Cognome']
 B \$Cognome = \$_POST[Cognome]
 C \$Cognome = \$_POST[\$Cognome]
 D Cognome = \$_POST['Cognome']
 10. Quale delle seguenti istruzioni visualizza in una pagina web un nome acquisito da un form HTML?
- A echo "nome = \$Nome"
 B echo "nome = " \$Nome
 C echo "nome = + \$Nome
 D echo "nome = Nome"
11. L'istruzione \$_SESSION['cognome'] = "Rossi" permette di:
 A impostare la variabile di sessione Rossi.
 B impostare il valore Rossi nella variabile "cognome".
 C prelevare il valore Rossi dalla variabile "cognome".
 D restituire il valore Rossi.

PREPARATI PER IL COLLOQUIO ORALE

1. Da chi viene elaborato il codice PHP? [vedi lez. 1]
2. Come viene definita una variabile in PHP? [vedi lez. 2]
3. Come avviene il richiamo di una pagina PHP da una pagina HTML? [vedi lez. 3]
4. Quali sono le modalità per mantenere i dati nel Web? [vedi lez. 3]
5. Come si esegue una query dopo aver stabilito una connessione? [vedi lez. 4]
6. Come avviene la connessione con un database? [vedi lez. 4]
7. Come avviene una query di aggiornamento? [vedi lez. 5]
8. Come si importano/esportano dati? [vedi lez. 7]



CLIL – IN ENGLISH, PLEASE

**AUDIO**Ascolta la pronuncia
del testo**ABSTRACT****PHP & MySQL**

The Client/Server model is the model used for the exchange of information. The term server refers to a computer that provides services, whereas a client is a computer that requests the services. Each of these two components can be static or dynamic, when it is able to process information by activating the necessary programs. Client-side technologies are characterized by the fact that the execution and interpretation of statements is carried out locally on the computer making the request to the server. Plug-ins, Java applets, and statements in scripting languages embedded in HTML programs are used. Server-side programming includes the set of techniques that permit the creation of dynamic web sites and web applications. The key component is the ability to access databases through web interfaces. The main languages and technologies that can be used server side are: Java Servlet, JSP, ASP.NET, and PHP. PHP is a scripting language used to develop dynamic web pages. Its syntax is based on that of the C language. The user can make pages interact with the MySQL database by using PHP-specific features and executing SQL commands on PHP pages.

The information that must be present in multiple dynamic pages of a site can be saved in cookies or session variables.

EXERCISES**True or false?**

1. The client cannot process information in the Client/Server model. T F
2. Applets are programs run by the client. T F
3. JavaScript is a scripting language. T F
4. PHP creates static pages. T F
5. A PHP program is executed by the browser. T F
6. A session variable does not expire. T F

Multiple choice

7. Cookies are:
 - A files stored on a server.
 - B files stored on a client.
 - C dynamic pages.
 - D static pages.
8. A session variable:
 - A expires when the user decides.
 - B does not expire.
 - C expires when the user closes the browser.
 - D expires in two hours.
9. PHP:
 - A is a language used to solve management problems.
 - B is a scripting language.
 - C cannot be used with MySQL.
 - D cannot be used with servers.

GLOSSARY

Applet: small application that performs one specific task that runs inside the client.

Cookie: a small piece of data sent from a website and stored in a user's web browser while the user is browsing that website. Also known as an HTTP cookie, web cookie, or browser cookie.

MySQL: is a popular choice of database used in web applications.

Scripting language or script language: a programming language that supports the writing of scripts.

Scripts: programs written for a special runtime environment that can interpret and automate the execution of tasks which could alternatively be executed one-by-one by a human operator.

**GLOSSARIO
CLIL**

COMPETENZE DISCIPLINARI

- Identificare e applicare le metodologie e le tecniche della gestione per progetti.
- Redigere relazioni tecniche e documentare le attività individuali e di gruppo relative a situazioni professionali.
- Interpretare i sistemi aziendali nei loro modelli, processi e flussi informativi con riferimento alle differenti tipologie di imprese.

**COMPETENZE
DEL XXI SECOLO****Competenze trasversali****PENSIERO CRITICO**

Saper analizzare e valutare situazioni in modo da impiegare informazioni e idee per formulare risposte e soluzioni.

COLLABORAZIONE

Saper lavorare in gruppo in vista di un obiettivo comune, prevedendo ed eventualmente gestendo conflitti.

Qualità caratteriali**CURIOSITÀ**

Inclinazione a porre domande con una mentalità aperta.

OBIETTIVI FORMATIVI

- Progettare la struttura di un sito e implementarne una parte.
- Consultare fonti Internet.
- Esporre i risultati della ricerca alla classe.

TEMPI

- Definizione della struttura del sito: 1 ora.
- Implementazione di una parte: 2 ore.
- Presentazione dei risultati e dibattito in classe: 1 ora.
- Autovalutazione: 10 minuti.

STRUMENTI

- Libro di testo.
- Dispositivo connesso a Internet.
- Foglio di carta
- Ambiente di sviluppo, PowerPoint.
- Proiettore collegato al computer in classe o in laboratorio.

IL TEMA

Di seguito è riportato un estratto della prima parte della **prova di informatica** dell'Esame di Stato del 2015 per l'Istituto Tecnico, settore Tecnologico, indirizzo Informatica e Telecomunicazioni.

Abbiamo presentato il testo del tema al termine delle unità 2 e 3. Abbiamo preso in considerazione i punti 1 e 2 al termine dell'unità 2, il punto 3 al termine dell'unità 3 e i punti 4 e 5 al termine dell'unità 4.

In questa Unità prendiamo in considerazione i punti 6 e 7.

PROGETTO E CODIFICA

[...] (Puoi vedere il testo del tema alla p. 66).

Il candidato, fatte le opportune ipotesi aggiuntive, sviluppi:

[...]

6. *il progetto della pagina dell'interfaccia web che permetta a un utente registrato di svolgere le operazioni specificate:*

Ricordiamo che: *il sito della community offre a tutti ... la consultazione dei dati online, tra cui: visualizzazione degli eventi di un certo tipo in ordine cronologico, con possibilità di filtro per territorio di una specifica provincia; visualizzazione di tutti i commenti e voti relativi a un evento;*

7. *la codifica in un linguaggio a scelta di un segmento significativo dell'applicazione web che consente l'interazione con la base di dati.*

[...]

I nuclei tematici fondamentali e gli obiettivi

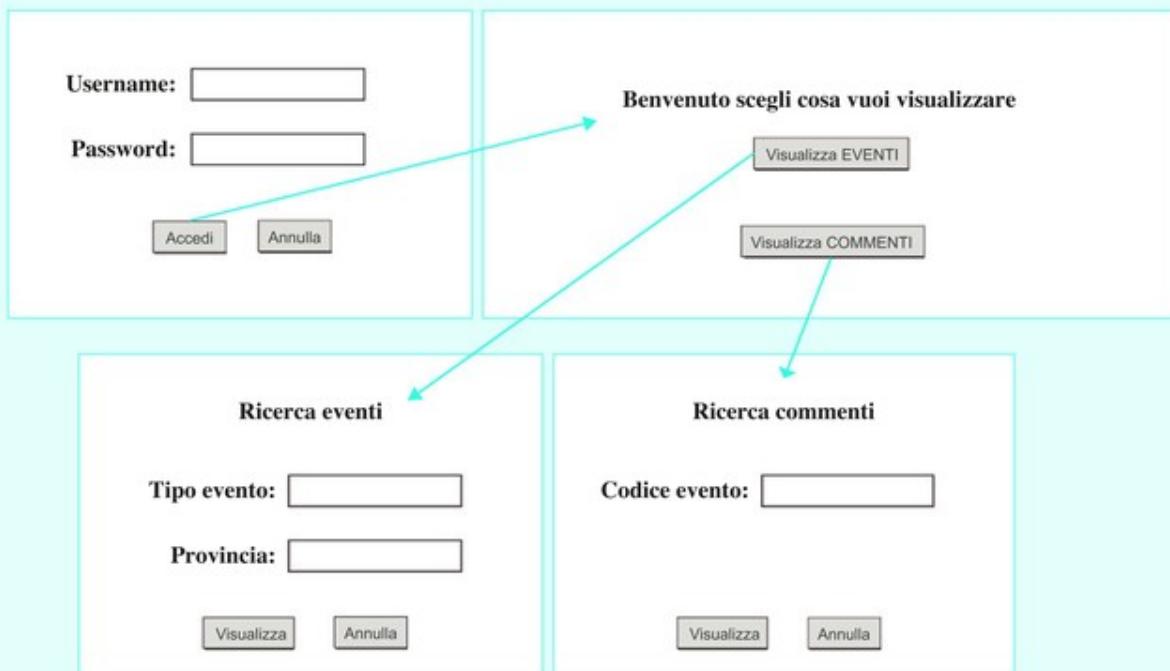
Fra i nuclei tematici fondamentali e gli obiettivi cui si riferiscono le prove d'esame (vedi la sezione finale "La preparazione alla seconda prova scritta dell'esame di Stato"), questo estratto si riferisce ai seguenti.

Nuclei tematici fondamentali	Obiettivi
<ul style="list-style-type: none"> Linguaggi per basi di dati: creazione, manipolazione e interrogazione di una base di dati. Tecnologie per il Web: i linguaggi lato client e lato server; realizzazione di applicazioni web anche con interfacciamento a basi di dati. 	<ul style="list-style-type: none"> Affrontare situazioni problematiche, utilizzando adeguate strategie cognitive e procedure operative orientate alla progettazione di soluzioni informatiche. Sviluppare applicazioni e servizi informatici per reti locali o geografiche. Scegliere sistemi e strumenti idonei al contesto proposto, in base alle loro caratteristiche funzionali. Realizzare progetti secondo procedure consolidate e criteri di sicurezza.

Svolgimento

Per quanto riguarda il punto 6 del tema d'esame possiamo progettare un sito web contenente una pagina iniziale che permetta a un utente registrato di accedere al sito. Se l'accesso va a buon fine, si apre una seconda pagina, in cui sarà possibile scegliere il tipo di visualizzazione cui si è interessati (visualizzazione degli eventi o dei commenti relativi a un dato evento). A seconda della visualizzazione scelta, si attiva un'altra pagina, dove si possono inserire i dati di input e visualizzare i risultati.

La struttura del sito sarà la seguente:



Per la codifica relativa al login si rimanda alla Lezione 6 di questa Unità; riportiamo la codifica relativa alla visualizzazione dei commenti di un dato evento.



CODIFICA

Visualizza.html

```
<html><head></head>
<body>
<p align = "center"><font size = '5'><strong>Ricerca commenti<br><br><br></p>
<form method = 'post' action = 'codice.php'>
<p align = "center"><font size = '5'>
Codice evento: <input type = 'text' name = 'codice' size = '10'>
<br><br><br>
<input type = 'submit' style="height:50px;" value = 'Visualizza' > &nbsp &nbsp &nbsp
<input type = 'reset' style="height:50px;" value = 'Annulla'>
</p>
</form>
</body></html>
```

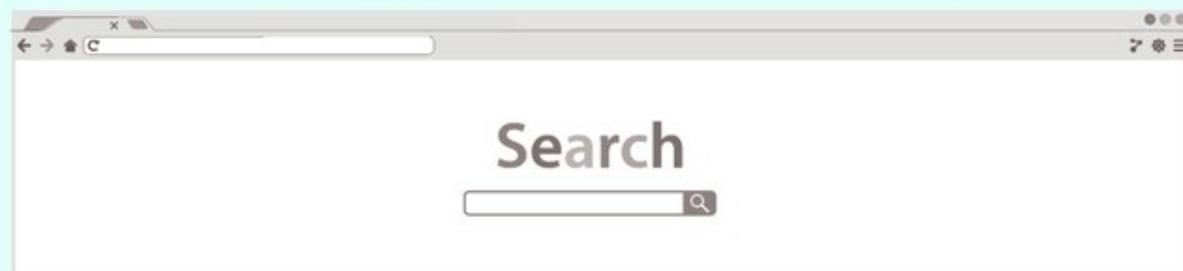
Visualizza.html

```
<?php
$hostname ="localhost";
$username = "root";
$password = "";
$dbname = "esame2015";
//connessione al server sql
$conn = mysqli_connect($hostname, $username, $password,$dbname);
if(!$conn){
    print "errore nella connessione";
    exit();
}
//recupera i dati passati dal form html
$codice = $_POST['codice'];
$query = "Select * from commenta where cod_evento = '$codice'";
$resultado= mysqli_query($conn,$query);
if (!$risultato){
    print "errore nel comando";
    exit();
}
$riga = mysqli_fetch_array($risultato);
if($riga){
    while($riga) {
        print "voto: ". $riga['voto']."&nbsp &nbsp &nbsp";
        print "data: ".$riga['data'].".&nbsp &nbsp &nbsp";
        print "commento: ".$riga['commento'].".<br>";
        $riga = mysqli_fetch_array($risultato);
    }
}
else{
    print "Attenzione!!! codice evento $codice non presente";
}
mysqli_close($conn);
?>
```

COMPITI DI REALTÀ

Dopo aver confrontato la risoluzione del tema d'esame con la tua, in gruppo svolgi le seguenti attività.

- COLLABORAZIONE** Effettuate una ricerca su Internet sullo stato dell'arte dei **sistemi di gestione degli eventi** in generale e, in particolare, di quelli legati alle **esibizioni musicali**. Noterete probabilmente che molte realtà sono orientate alla vendita di biglietti per gli eventi. Esiste però anche un'applicazione, denominata **Local** e collegata a Facebook, che ne ha rinnovato il potenziale e soprattutto gli utenti. Analizzate la struttura e il funzionamento di questa applicazione.
- PENSIERO CRITICO** Individuate quelle che, secondo voi, sono le potenzialità dalla app Local, poi evidenziate anche i limiti (per esempio, un limite è l'interazione molto stretta con le attività svolte su Facebook, che potrebbe dare all'utente la sensazione che la app lavori senza un suo input diretto).
- COLLABORAZIONE** Raccogliete i risultati della vostra ricerca e della vostra discussione in una presentazione in PowerPoint (o in un altro strumento analogo), formata al massimo da cinque slide.
- COMUNICAZIONE** In classe, con la supervisione dell'insegnante, condividete con i compagni la presentazione realizzata dal vostro gruppo: confrontate le tematiche emerse e discutetene.
- PENSIERO CRITICO** Infine, in classe, discutete con i compagni lo svolgimento utilizzato per risolvere i quesiti del tema d'esame; poi proponete eventuali modifiche, che vi sembrino più adeguate.



AUTOVALUTAZIONE

Al termine delle attività rifletti sull'esperienza e completa la tabella di autovalutazione.

Attività	LIVELLO		
	Base	Intermedio	Avanzato
Ho compreso senza difficoltà le richieste dell'attività proposta?	Con la guida dell'insegnante e dei compagni ho compreso quasi tutte le richieste. <input type="checkbox"/>	Ho compreso le richieste e in parte le ho svolte autonomamente. <input type="checkbox"/>	Ho identificato le richieste e le ho svolte senza difficoltà. <input type="checkbox"/>
Sono riuscito a progettare il sito web?	Ho progettato il sito in modo poco dettagliato e descrivendo gli elementi in modo sommario. <input type="checkbox"/>	Ho progettato il sito tralasciando alcuni particolari. <input type="checkbox"/>	Ho realizzato uno schema completo e ben descritto del sito con le interazioni presenti tra le pagine. <input type="checkbox"/>
Sono riuscito a produrre la codifica di una parte significativa del codice?	Ho realizzato la parte HTML, ho iniziato a scrivere la codifica PHP, ma non sono riuscito a completarla. <input type="checkbox"/>	Ho realizzato la codifica di entrambe le parti, ma ci sono degli errori nell'esecuzione. <input type="checkbox"/>	Ho realizzato la codifica del sito, prelevando i dati dal database e ottenendo i risultati desiderati. <input type="checkbox"/>

7

Database in rete - ASP.NET



ESERCIZI COMMENTATI

Segui la risoluzione passo passo



LABORATORI CASE STUDIES

Approfondisci con i casi pratici



PREREQUISITI

- Conoscere le caratteristiche del modello Client/Server.
- Conoscere gli elementi di base di HTML e di C#.
- Conoscere i comandi SQL.

PER COMINCIARE

1. Per inserire una casella di testo in una pagina HTML si usa il tag:
 A <input type="text">
 B <input type="radio">
 C <input type="checkbox">
 D <select name>
2. Data la tabella Elettore (Id_Elettore, DatiAnagrafici, AnnoNascita, Seggio) il comando `select count (*) from Elettore where AnnoNascita > 1980` fornisce:
 A i dati degli elettori nati prima del 1980
 B i dati degli elettori nati dopo il 1980
 C il numero degli elettori
 D il numero degli elettori nati dopo il 1980
3. Facendo riferimento alla tabella del quesito precedente, scrivi la query per estrarre tutti i seggi che hanno meno di 200 elettori.



CONOSCENZE

- Conoscere le possibilità di programmazione web.
- Conoscere le caratteristiche della programmazione lato client e lato server.
- Conoscere la programmazione ASP.NET.



ABILITÀ

- Saper creare pagine dinamiche con ASP.NET.
- Saper realizzare il passaggio di informazioni tra pagine web.
- Saper accedere al database e reperire le informazioni.
- Saper gestire un login di connessione.
- Saper manipolare i dati di un database.



COMPETENZE

- Scegliere dispositivi e strumenti per sviluppare applicazioni informatiche per reti locali o servizi a distanza.
- Utilizzare i sistemi informativi aziendali e gli strumenti di comunicazione integrata d'impresa, per realizzare attività comunicative con riferimento a differenti contesti.

1. HTML E ASP.NET

Le pagine .Aspx sono create in modo simile alle pagine web HTML statiche, che non includono l'elaborazione lato server, ma contengono elementi supplementari riconosciuti ed elaborati da ASP.NET durante l'esecuzione della pagina.

Le caratteristiche che distinguono le pagine web ASP.NET dalle pagine statiche HTML (o di altro tipo) sono le seguenti:

- contengono la direttiva @ Page, che descrive il tipo di pagina che si desidera creare;
- presentano un elemento *form*, attraverso il quale il client è in grado di inviare i dati al server;
- contengono i controlli server web (caselle di testo, buttoni, etichette...), per interagire con il server.

La pagina .Aspx inizia con la direttiva @ Page:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
```

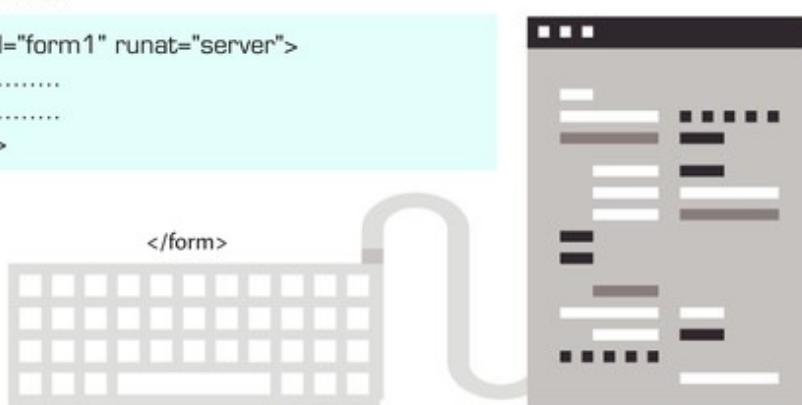
che consente di specificare numerose opzioni di configurazione per la pagina, tra cui:

- **Language**: specifica il linguaggio di programmazione server per il codice nella pagina, C# nel nostro caso;
- **AutoEventWireup**: indica se ASP.NET può intercettare automaticamente i comuni eventi di pagina come Load;
- **CodeFile**: indica il nome del file .cs che contiene il codice associato alla pagina.

■ IL FORM

Se nella pagina sono presenti controlli che consentono all'utente di interagire con la pagina e di inoltrarla al server, questa deve includere nel body un elemento **form**: la pagina può contenere solo un elemento form che deve contenere l'attributo **runat** con il valore impostato su **server**. Questo attributo consente di creare un riferimento tra il form e il codice C# lato server.

```
<form id="form1" runat="server">  
.....  
.....  
</form>
```



Elementi del form

Il form contiene gli elementi che saranno presentati all'utente. Ogni elemento ha sempre gli attributi:

- **runat="server"**, per rendere disponibile l'elemento al codice lato server;
- **ID="nomeElemento"**, per identificare l'elemento.

I più comuni controlli server web sono:

- **asp:Button**: bottone a cui risponde per default l'evento onClick lato server;
- **asp:TextBox**: casella di testo per l'inserimento dei dati;
- **asp:CheckBox**: casella di scelta a selezione multipla;
- **asp:DropDownList**: la classica selectbox, conosciuta anche come combobox o menu a tendina;
- **asp:RadioButton**: casella di scelta a selezione singola.

Button

L'oggetto **Button** crea un bottone cliccabile:

```
<asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Inserisci" />
```

Le proprietà per l'oggetto Button sono:

- **OnClick**: indica la funzione lato server che sarà eseguita al click sul bottone;
- **OnClientClick**: indica la funzione lato client che sarà eseguita al click sul bottone;
- **Text**: specifica il testo del bottone.

Al click sul bottone il server esegue il codice presente nella funzione:

```
protected void Button1_Click(object sender, EventArgs e)
{
    .....
}
```

TextBox

Un controllo **TextBox** accetta l'input dell'utente:

```
<asp:TextBox ID="TxtNome" runat="server"></asp:TextBox>
```

Le proprietà per l'oggetto TextBox sono:

- **Columns**: specifica la larghezza della textBox in termini di caratteri;
- **MaxLength**: numerico, specifica il massimo numero di caratteri che possono essere digitati nella casella di testo;
- **ReadOnly**: booleano, specifica se sulla casella è possibile scrivere o meno;
- **Rows**: specifica il numero di righe della casella ed è funzionante solo nel caso che la proprietà **TextMode** sia impostata su "MultiLine": in sostanza si creerà una textarea;
- **Text**: specifica il valore di default della textbox;
- **TextMode**: specifica alcune caratteristiche della textbox; può assumere i valori SingleLine (di default se non specificato), MultiLine (come sudetto) e Password (cripta i caratteri digitati nella textbox).

Nella pagina che contiene il codice (che ha estensione .aspx.cs) il dato contenuto nella TextBox è recuperato dalla proprietà Text della TextBox:

```
nomeVariabile = TxtNome.Text;
```

Il valore contenuto nella proprietà Text è di tipo stringa e quindi saranno necessarie le funzioni di conversione per ottenere valori numerici: (Numero = Int.Parse(Txt.Numero.Text)).

CheckBox

Il controllo CheckBox crea una casella di controllo, che consente all'utente di passare da uno stato true a uno stato false e viceversa:

```
<asp:CheckBox ID="CheckBox1" runat="server" />
```

Le proprietà per l'oggetto TextBox sono:

- **Checked:** booleano, specifica se un item deve essere selezionato per default o meno;
- **Text:** imposta il testo al CheckBox.

Nella pagina che contiene il codice per verificare se il controllo è stato selezionato, si scrive:

```
if (CheckBox1.Checked) ....
```

RadioButton

L'oggetto RadioButton crea un pulsante radio.

```
<asp:RadioButton ID="RadioButton1" runat="server" />
```

Le proprietà per l'oggetto RadioButton sono:

- **Checked:** specifica se un item deve essere selezionato per default o meno (ovviamente, la selezione di uno esclude quella dell'altro);
- **GroupName:** indica il gruppo di appartenenza dell'oggetto;
- **Text:** imposta il testo al radioButton.

Nella pagina che contiene il codice (che ha estensione .aspx.cs) per verificare se il controllo è stato selezionato, si scrive:

```
if (RadioButton1.Checked) ....
```

Nelle applicazioni si può presentare il caso che chi progetta il form non conosca i valori associati ai RadioButton, perché questi ultimi sono definiti in fase di esecuzione. Può allora essere utile estendere il controllo RadioButton, ampliandolo con il controllo RadioButtonList.

In fase di esecuzione si creano i radioButton contenuti in RadioButtonList con il comando:

```
RadioButton1.Items.Add("valore")
```

e il valore del radioButton selezionato è prelevato con:

```
string = RadioButton1.SelectedValue
```

DropDownList

Il controllo server web **DropDownList** consente all'utente di selezionare un unico elemento da un menu a discesa predefinito.

```
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
    OnSelectedIndexChanged=" DropDownList1_SelectedIndexChanged">
    <asp:ListItem>valore1</asp:ListItem>
    <asp:ListItem>Giallo</asp:ListItem>
    .....
</asp:DropDownList>
```

Affinché sia intercettato l'evento `SelectedIndexChanged`, la proprietà `AutoPostBack` deve essere impostata a true, per poter inviare al server la pagina quando l'utente ha effettuato la sua scelta sulla `DropDownList`. La voce selezionata è catturata con:

```
string = DropDownList.SelectedItem.Text;
```

L'esempio seguente mostra come stampare il colore selezionato scelto in una `DropDownList`.

Codifica pagina Aspx:

```
<html>
<body>
<form runat="server">
    <asp:DropDownList ID="DD1" runat="server" AutoPostBack="True"
        OnSelectedIndexChanged="DD1_SelectedIndexChanged">
        <asp:ListItem>Rosso</asp:ListItem>
        <asp:ListItem>Giallo</asp:ListItem>
        <asp:ListItem>Verde</asp:ListItem>
        <asp:ListItem>Blu</asp:ListItem>
    </asp:DropDownList>
    <br />
    <br />
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    <br />
</form>
</body>
</html>
```

Codifica pagina Aspx.cs:

```
public partial class _Default : System.Web.UI.Page
{
    protected void DD1_SelectedIndexChanged(object sender, EventArgs e)
    {
        Label1.Text = "Hai scelto il colore " + DD1.SelectedItem.Text;
    }
}
```

Calendar

Una componente ASP.NET molto usata è l'oggetto **Calendar**, che è in grado di generare con pochi comandi un utile calendario. L'esempio di seguito riportato ne implementa una versione molto semplice:

Codifica pagina Aspx:

```
<form runat="server">

    <asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
    <br />
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
</form>
```

Codifica pagina Aspx.cs:

```
public partial class _Default : System.Web.UI.Page
{
    protected void Calendar1_SelectionChanged(object sender, EventArgs e)
    {
        Label1.Text = Calendar1.SelectedDate.ToShortDateString();
    }
}
```

L'evento Page_Load

Quando la pagina viene caricata, ASP.NET chiama automaticamente il **Page_Load** e ne esegue il codice:

```
<script runat="server">
Sub Page_Load
Label1.Text = "The date and time is " + DateTime.Now;End Sub
</script>

<html>
<body>
<form runat="server">
<h3>
    <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text
    ="Button" />
</h3>
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
</form>
</body>
</html>
```

Bisogna tenere presente che la pagina è caricata ogni volta che si clicca sul bottone e, pertanto, tutte le volte viene eseguito il codice associato all'evento **Page_Load**.

Il PostBack

Quando una pagina di un sito web è richiamata da se stessa è necessario capire se si tratta della prima richiesta o se la pagina ha già effettuato un richiamo (post) a se stessa. Da questo fatto dipendono le operazioni successive. Prima di ASP.NET, nei form venivano usati dei campi nascosti (hidden) per poter riconoscere se si trattava della prima chiamata della pagina o di chiamate successive.

Questa tecnica è chiamata **postback**. Il postback è implementato in ASP.NET in modo invisibile al programmatore. La proprietà **Page.IsPostBack** mostra appunto se una pagina è stata caricata per la prima volta oppure in seguito a un postback.

Con il codice seguente è mostrato un semplice utilizzo della proprietà **IsPostBack**; si vuole stampare il messaggio "Primo caricamento" solo quando il form è caricato la prima volta:

```
if (!IsPostBack)
    Label2.Text = "Primo caricamento";
else
    Label2.Text = "Caricamento successivo";
```

LABORATORIO

IL PROBLEMA Crea un'applicazione ASP.NET per visualizzare il quadrato e il cubo di un numero inserito in input.

L'ANALISI Supponiamo di aver creato una applicazione ASP.NET seguendo i passi illustrati nella Lezione.

Prepariamo adesso la pagina .aspx che contiene la casella di testo dove andremo a inserire il numero da elaborare.

L'INTERFACCIA

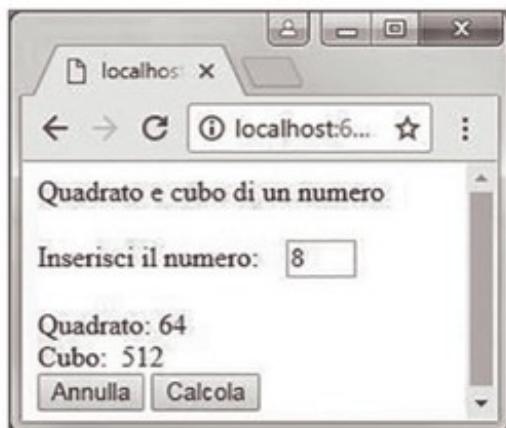


```

Dim q As Integer
Dim c As Integer
If txtNum.Text = "" Then
    lblQuadrato.Text = "Inserire un numero"
Else
    num = txtNum.Text
    q = num * num
    lblQuadrato.Text = q
    c = num * num * num
    lblCubo.Text = c
End If
End Sub
Protected Sub btnAnnulla_Click(...) Handles btnAnnulla.Click
    lblQuadrato.Text = ""
    lblCubo.Text = ""
    txtNum.Text = ""
End Sub
End Class

```

IL RISULTATO



FISSA LE CONOSCENZE

1. Quale clausola deve essere impostata sulla pagina HTML affinché venga eseguito il codice associato a un bottone?
2. Come si verifica quale RadioButton è stato selezionato?
3. Come si individua quale valore è stato selezionato da una DropDownList ?
4. Quando viene eseguito il PageLoad?
5. Che cosa è il postback?

2. PASSAGGIO PARAMETRI

Nella costruzione di un sito web spesso è necessario realizzare più pagine che si richiamano tra loro e che trasferiscano informazioni da una all'altra.

Il richiamo di una pagina avviene utilizzando il metodo Redirect della classe Response:

pagina.aspx è la pagina che viene richiamata.

```
Response.Redirect ("pagina.aspx")
```

In ASP.NET sono previste diverse possibilità per il passaggio di informazioni tra pagine che si richiamano. In questa Lezione vediamo nel dettaglio: il passaggio di informazioni tramite l'impostazione di una stringa di query, la memorizzazione delle informazioni in un cookie e l'uso delle variabili di sessione. Esiste inoltre la possibilità di passare lo stato della sessione e lo stato dell'applicazione, ma per questo si rimanda a manuali specifici.

L'impostazione di una stringa di query è il metodo più semplice per il passaggio di informazioni. È necessario l'inserimento all'interno dell'URL della pagina a cui si viene reindirizzati con la seguente sintassi:

```
Response.Redirect ("pagina.aspx?nomevar=valore")
```

Nella pagina *pagina.aspx* la variabile *nomevar* assumerà il valore specificato nel richiamo, utilizzando il metodo *QueryString* dell'oggetto *Request*:

```
nomevar = Request.QueryString("nomevar")
```

Lo stesso obiettivo può essere raggiunto utilizzando un file cookie salvato automaticamente nella memoria del browser, oppure sul disco dell'utente se dovrà contenere informazioni di lunga durata (la durata del cookie può essere definita con il metodo *Expires*; nel caso non sia definita, il cookie viene considerato temporaneo).

La sintassi per la dichiarazione di un cookie è la seguente:

nomecookie indica il nome dell'istanza e nome indica il nome fisico del file cookie.

```
HttpCookie nomecookie = new HttpCookie ("nome")
```

Per inserire una variabile con un dato valore nel cookie si usa la sintassi:

```
Response.Cookies["nomevar"].value = valore
```

È poi necessario aggiungere il cookie creato alla **collezione dei cookie** della pagina web, usando il metodo *Add*:

```
Response.Cookies.Add (nomecookie)
```

Per leggere un cookie viene usato il comando *Request.Cookies* secondo la seguente sintassi:

```
nomevar = Request.Cookies ("nomevar")
```

Se vogliamo creare delle pagine che ricordino delle informazioni, dobbiamo prima memorizzarle. Per farlo possiamo usare le sessioni che

utilizzano i cookie in modo trasparente al programmatore. L'oggetto Session è utilizzato per memorizzare o cambiare le informazioni circa la sessione di un singolo utente. Le variabili memorizzate nell'oggetto Session sono disponibili per tutte le pagine dell'applicazione.

L'esempio successivo spiega come memorizzare delle **variabili di sessione**. Imposteremo la variabile di sessione username a "Mario" e la variabile age a 20:

```
Session("username") = "Mario"  
Session("age") = 20
```

Le variabili impostate possono essere recuperate in qualsiasi parte dell'applicazione con le istruzioni:

```
username = Session("username")  
age = Session("age")
```

Per rimuovere tutte le variabili di sessione si può utilizzare il metodo RemoveAll della collezione Contents, che le contiene tutte:

```
Session.Contents.RemoveAll()
```

Oppure possono essere ripulite con l'istruzione:

```
Session("username") = ""
```

Una sessione termina, se un utente non ha richiesto pagine o non ha fatto il refresh in una pagina dell'applicazione per un determinato lasso di tempo. Di default questo tempo è 20 minuti.

Se si vuole cambiare l'intervallo di default, occorre impostare la proprietà **Timeout**.

```
Session.Timeout=60
```

Per **terminare una sessione immediatamente**, può utilizzare il metodo **Abandon**:

```
Session.Abandon
```

LABORATORIO

IL PROBLEMA Costruisci una pagina in cui, dopo aver richiesto il nome e cognome di un utente, venga richiamata un'altra pagina all'interno della quale viene visualizzato un messaggio di benvenuto, utilizzando i vari tipi di passaggio di parametri.

L'ANALISI Costruire una pagina in cui siano presenti due caselle di testo, in cui l'utente deve inserire, rispettivamente, il nome e il cognome e tre bottoni associati. Ai tre bottoni si aggiunge un controllo sulle caselle di testo e il

richiamo a un'altra pagina in cui, al momento del caricamento, in una label sarà inserito il contenuto delle variabili preceduto da un messaggio di benvenuto.

L'INTERFACCIA

Nome:	<input type="text"/>	Cognome:	<input type="text"/>
<input type="button" value="Stringa"/>	<input type="button" value="Cookie"/>	<input type="button" value="Session"/>	

CODIFICA DEFAULT.ASPX

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>
<form id="form1" runat="server">
<div class="auto-style1">
<table>
<tr>
<td>Nome: </td>
<td><asp:TextBox ID="TxtNome" runat="server" CssClass="input" Width="75px"></asp:TextBox></td>
<td> Cognome: </td>
<td><asp:TextBox ID="TxtCognome" runat="server" CssClass="input" Width="75px"></asp:TextBox></td>
</tr>
<tr>
<td> <br />
<asp:Button ID="BtnStringa" runat="server" Text="Stringa" /> </td>
<td><br />
<asp:Button ID="BtnCookie" runat="server" Text="Cookie" /> </td>
<td><br />
<asp:Button ID="BtnSession" runat="server" Text="Session" /> </td>
</tr>
</table>
</div>
</form>
</body>
</html>
```

CODIFICA C#.NET DEFAULT.ASPX.CS

```
using System;

public partial class _Default : System.Web.UI.Page
{
    protected void BtnStringa_Click(object sender, EventArgs e)
    {
        String stringa;
        if((TxtNome.Text != "") || (TxtCognome.Text != ""))
        {
            stringa = TxtNome.Text + " " + TxtCognome.Text;
            Response.Redirect("Stringa.aspx?nome=" + stringa);
        }
    }

    protected void BtnCookie_Click(object sender, EventArgs e)
```

```

{
    if ((TxtNome.Text != "") || (TxtCognome.Text != ""))
    {
        HttpCookie MioCook = new HttpCookie("ASPNET");
        Response.Cookies["nome"].Value = TxtNome.Text + " " +
            TxtCognome.Text;
        Response.Cookies.Add(MioCook);
        Response.Redirect("Cookies.aspx");
    }
}

protected void BtnSession_Click(object sender, EventArgs e)
{
    if ((TxtNome.Text != "") || (TxtCognome.Text != ""))
    {
        Session["nome"] = TxtNome.Text;
        Session["cognome"] = TxtCognome.Text;
        Response.Redirect("Session.aspx");
    }
}
}

```

CODIFICA C#.NET STRING.ASPX.CS

```

using System;
public partial class Stringa : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        lblMessaggio.Text = "Benvenuto in questa pagina " +
            Request.QueryString["nome"];
    }
}

```

C#.NET COOKIES.ASPX.CS

```

using System;
public partial class Cookies : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        lblMessaggio.Text = "Benvenuto in questa pagina " +
            Request.Cookies["nome"].Value;
    }
}

```

CODIFICA C#.NET SESSION.ASPX.CS

```
Partial Class Default2  
    Inherits System.Web.UI.Page  
    Protected Sub Page_Load(sender As Object, e As EventArgs) Handles Me.Load  
        LblMessaggio.Text = "Benvenuto in questa pagina " & Session("cognome") & " " &  
        Session("nome")  
    End Sub  
End Class
```

IL RISULTATO

Selezionando il bottone Stringa si visualizza la pagina della **fig. 1**, selezionando bottone Cookie, si visualizza la pagina della **fig. 2**, mentre si visualizza la pagina della **fig. 3** selezionando il bottone Sessione.

fig. 1



fig. 2



fig. 3



FISSA LE CONOSCENZE

1. Quale comando richiama una pagina ASP?
2. A che cosa serve il comando Request.QueryString?
3. Che cos'è un cookie?
4. Quando si usa la variabile di sessione?
5. A che cosa serve la proprietà timeout?

3. CONNESSIONE AL DATABASE

La maggior parte delle applicazioni web hanno la necessità di interrograre o aggiornare dati memorizzati in una base di dati. Per venire incontro a tale necessità, la piattaforma .NET fornisce tutte le componenti necessarie per creare ed eseguire le applicazioni che richiedono di scambiare dati su Internet.

Questa piattaforma, il **.NET Framework**, include **ADO.NET**, che è un insieme di classi che, basandosi sulla tecnologia **OLEDB.NET**, consentono ai programmatore di interagire con le origini dati.

OLEDB.NET è il metodo Microsoft per l'accesso a fonti di dati di diversa natura, quali database relazionali, messaggi di posta elettronica, pagine web, fogli elettronici di Excel e così via.

OLEDB.NET accede alle singole fonti dati attraverso un data provider (fornitore di dati), che ha caratteristiche diverse in funzione della fonte di dati utilizzata (per esempio, esistono provider diversi per database Access o per SQL Server o per Oracle).

■ Connessione con un database ACCESS

Per connettere un'applicazione a un'origine dati di Microsoft Office Access, il provider da utilizzare dipende dalla versione di Access in cui è stata creata l'origine dati:

- **Access 2003** e versioni precedenti: il pacchetto richiede il provider OLEDB Microsoft Jet (Microsoft.Jet.OLEDB.4.0);
- **da Access 2007** in poi: il pacchetto richiede il provider OLEDB per il modulo di gestione di database di Microsoft Office 12.0 Access (Microsoft.ACE.OLEDB.12.0).

Per evitare di scrivere per esteso i nomi dei metodi e degli attributi delle classi utilizzate, è necessario inserire all'inizio della codifica C#.NET (prima riga del file sorgente) la seguente istruzione:

```
using System.Data.OleDb;
```

Per creare una connessione con un database relazionale, si utilizza un oggetto della classe **OleDbConnection**:

```
OleDbConnection connection = new OleDbConnection();
```

Attraverso l'attributo **ConnectionString** dell'oggetto **OleDbConnection** è possibile fornire le informazioni necessarie per connettersi al database: il tipo di provider utilizzato, il percorso fisico del drive su cui è memorizzato il database (Data Source) e, in maniera opzionale, la modalità di accesso, il nome utente e la password eventualmente richiesti per l'accesso ai dati.

```
ConnectionString="Provider=Microsoft.ACE.OLEDB.12.0; Data Source='Maggazzino.accdb'"
```

In questo caso il database si trova sul server nella stessa directory dell'applicazione. È però consigliabile evitare di inserire percorsi assoluti nella connectionString e, pertanto, risulta più conveniente salvare il database nella directory App_Data del sito web e specificare nel DataSource il percorso `|Data|Directory|nomedb`.

Nell'esempio precedente avremmo:

```
ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0";& Data Source=|DataDirectory|"Magazzino.accdb"
```

Per esempio, per connettersi al database Magazzino di Microsoft Office Access 2007 si ha:

```
ConnessioneDb.ConnectionString. = "Provider=Microsoft.ACE.OLEDB.12.0;"& "Data Source= "Magazzino"
```

I metodi più utilizzati dell'oggetto OleDbConnection sono:

```
ConnessioneDb.Open();
```

utilizzato per aprire una connessione tra il database e l'applicazione;

```
ConnessioneDb.Close();
```

utilizzato per chiudere la connessione quando si è terminato di utilizzare il database;

```
ConnessioneDb.State();
```

restituisce il valore 0 (zero) se la connessione non è aperta, 1 (uno) in caso contrario.

■ Connessione con un database SQL Server

Per connettere un'applicazione C#.NET a SQL Server è necessario importare la libreria System.Data.SqlClient che contiene le classi per gestire il database:

```
using System.Data.SqlClient;
```

Successivamente, per eseguire la connessione si utilizza un'istanza della classe SqlConnection:

```
SqlConnection ConnessioneDB = new SqlConnection();
```

Per creare una connessione con un database relazionale, si utilizza un oggetto della classe **SqlConnection**:

```
Dim ConnessioneDb As New SqlConnection
```

Attraverso l'attributo `ConnectionString` dell'oggetto `SqlConnection` è possibile fornire le informazioni necessarie per connettersi al database, cioè il tipo di provider utilizzato, il percorso fisico del drive su cui è memorizzato il database (Data Source) e, in maniera opzionale, la modalità di accesso, il nome utente e la password eventualmente richiesti per l'accesso ai dati.

Per esempio per connettersi al database Magazzino si ha:

```
connDb.ConnectionString = "Data Source = (LocalDB)\MSSQLLocalDB;  
Initial Catalog= Magazzino";
```

dove:

(localDB)\MSSQLLocalDB è l'istanza di SQL Server installata sul computer, che si può ottenere lanciando Visual Studio, e in *Visualizza > Esplora oggetti di SQL Server* (fig. 4) si individua l'istanza di SQL Server (fig. 5)



fig. 4 Comando per individuare l'istanza di SQL Server



fig. 5 Nome dell'istanza di SQL Server

In questo caso il database si trova sul server nella stessa directory dell'applicazione. È però consigliabile evitare di inserire percorsi assoluti nella connectionString e, pertanto, risulta più conveniente salvare il database nella directory App_Data del sito web e specificare nel DataSource il percorso `|Data|Directory|nomedb`. In questo caso la stringa di connessione risulta più complessa:

```
ConnessioneDb.ConnectionString = "Data Source = "+  
"(LocalDB)\MSSQLLocalDB; AttachDbFileName = "+  
"|DataDirectory|magazzino.mdf; Initial Catalog = magazzino.mdf";
```

I metodi più utilizzati dell'oggetto SqlConnection sono:

```
ConnessioneDb.Open();
```

utilizzato per aprire una connessione tra il database e l'applicazione;

```
ConnessioneDb.Close();
```

utilizzato per chiudere la connessione quando si è terminato di utilizzare il database;

```
ConnessioneDb.State();
```

restituisce il valore 0 (zero) se la connessione non è aperta, 1 (uno) in caso contrario.

LABORATORIO

IL PROBLEMA Prepara una connessione con il database *magazzino.accdb*, utilizzando la tecnologia ADO.NET, e visualizza lo stato della connessione dopo l'apertura e dopo la chiusura.

L'ANALISI La connessione con la fonte dati prevede la definizione del provider e del nome del database a cui ci si vuole connettere. Per garantire la portabilità del progetto, bisogna inserire il database nella cartella App_Data del nostro progetto e indicare tale percorso con Source= |DataDirectory|.

LA VARIABILE

Nome	Tipo	Utilizzo	Descrizione
connDb	OleDbConnection	lavoro	connessione con il database

CODIFICA C#.NET (DATABASE ACCESS)

```
using System;
using System.Data.OleDb;
public partial class Connelli : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        OleDbConnection connDb = new OleDbConnection();
        connDb.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0; "+
        "Data Source=|DataDirectory|/magazzino.accdb";
        try
        {
            connDb.Open();
            Response.Write("Connessione avvenuta correttamente");
            connDb.Close();
        }
        catch
        {
            Response.Write("ERRORE - Connessione non corretta");
        }
    }
}
```



4. VISUALIZZAZIONE DATI

La visualizzazione dei dati avviene dopo aver definito una connessione con il database che, come abbiamo visto nel paragrafo precedente, è aperta con il metodo Open e viene chiusa al termine del programma stesso (metodo Close).

Dopo avere aperto la connessione, si dichiara una istanza della classe che rappresenta un'istruzione SQL da eseguire su un'origine dati:

```
OleDbCommand comandoSql = new OleDbCommand(); per database Access  
e
```

```
SqlCommand comandoSql = new SqlCommand(); per database SQL Server
```

Utilizzeremo le proprietà CommandText e Connection dell'oggetto appena istanziato e, precisamente, con la proprietà CommandText definiamo il comando SQL da eseguire:

```
comandosql.CommandText = "select ....";
```

mentre con l'attributo Connection andiamo a definire il nome della connessione da utilizzare:

```
comandosql.Connection = ConnessioneDb;
```

Il metodo Dispose() rilascia tutte le risorse usate dal comando SQL:

```
comandosql.Dispose();
```

Come abbiamo visto nelle lezioni precedenti, l'esecuzione di una query SQL fornisce una nuova tabella, anche se, come risultato, si ottiene una sola riga o un solo valore (Count, Avg, Sum, Min, Max). In ogni caso è necessario dichiarare nel programma un tipo di variabile in grado di contenere questa tabella e, per questo, si dichiara un dataReader:

```
OleDbDataReader rs;
```

← per database Access

```
e
```

```
SqlDataReader rs;
```

← per database SQL Server

A questo punto attraverso il metodo ExecuteReader() della classe DataReader si esegue il comando SQL specificato e si pongono i risultati nella tabella dichiarata con il DataReader:

```
rs.["nome_campo"]
```

È possibile utilizzare i dati salvati nel DataReader, utilizzando i seguenti metodi dell'oggetto OleDbDataReader:

```
rs.HasRows()
```

che restituisce False se la tabella non contiene alcuna riga e True altrimenti;

```
rs.Read()
```

che viene usato per ottenere una riga dai risultati della query e sposta la riga corrente della tabella in avanti di una posizione. Restituisce False se si va oltre l'ultima riga, True altrimenti;

```
rs.Item('nome_campo')
```

che restituisce il valore contenuto nel campo nome_campo;

```
rs.Close()
```

che chiude l'oggetto DataReader una volta terminato il suo uso.

In questa Lezione e in quelle successive si fa riferimento al database *Magazzino*, che contiene le seguenti tabelle:

UTENTI			
Nome campo	Descrizione	Tempo	Chiave
user	userId	testo	primaria
cognome	cognome	testo	
nome	nome	testo	
psw	password	testo	

PRODOTTI			
Nome campo	Descrizione	Tempo	Chiave
idProdotto	codice	testo	primaria
descrizione	descrizione	testo	
prezzo	prezzo unitario	numerico	
qta	quantità a magazzino	numerico	

Negli esercizi che seguono non viene riportata la codifica .aspx delle pagine, poiché viene generata automaticamente da Visual Studio.

LABORATORIO

IL PROBLEMA Visualizza i dati del prodotto corrispondente al codice inserito in input.

L'ANALISI Dopo aver stabilito la connessione con il database, fai eseguire la query che estrae dalla tabella Prodotti i dati del prodotto corrispondente al codice inserito in input (txtIdProdotto.Text). Se la query ha dato come risultato un insieme vuoto (rsProdotto.Read() = False) significa che il codice inserito non esiste, in caso contrario sono visualizzati i dati del prodotto.

L'INTERFACCIA

VISUALIZZA PRODOTTO

CODICE PRODOTTO:

Descrizione:

Prezzo:

Quantità:

CODIFICA C#.NET (DATABASE ACCESS)

```
using System;
using System.Data.OleDb;

public partial class _Default : System.Web.UI.Page
{
    protected void btnCerca_Click(object sender, EventArgs e)
    {
        OleDbConnection connection = new OleDbConnection();
        OleDbCommand command = new OleDbCommand();
        OleDbDataReader rsProdotto;
        lblDescrizione.Text = " ";
        lblPrezzo.Text = " ";
        lblQta.Text = " ";
        string idPr = txtIdProdotto.Text;

        connDb.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;" +
        "Data Source=|DataDirectory|/magazzino.accdb";
        connection.Open();
        string query = "SELECT * FROM Prodotti WHERE idProdotto = '" + idPr + "'";
        command.Connection = connection;
        command.CommandText = query;
        rsProdotto = command.ExecuteReader();
        if (rsProdotto.Read())
        {
            lblDescrizione.Text = rsProdotto["descrizione"].ToString();
            lblPrezzo.Text = rsProdotto["Prezzo"].ToString();
            lblQta.Text = rsProdotto["Qta"].ToString();
        }
        else
            lblErrore.Text = "codice prodotto inesistente";
        rsProdotto.Close();
        connection.Close();
    }
}
```

CODIFICA C#.NET (DATABASE SQL SERVER)

```
using System;
using System.Data.SqlClient;
public partial class _Default : System.Web.UI.Page
{
    protected void btnCerca_Click(object sender, EventArgs e)
    {
        SqlConnection connDb = new SqlConnection();
        SqlCommand command = new SqlCommand();
        SqlDataReader rsProdotto;
        lblDescrizione.Text = " ";
        lblPrezzo.Text = " ";
        lblQta.Text = " ";
        string idPr = txtIdProdotto.Text;
        connDb.ConnectionString = "Data Source = [LocalDB]\MSSQLLocalDB; " +
```

```

    " AttachDbFileName =|DataDirectory|magazzino.mdf; " +
    "Initial Catalog = magazzino.mdf";
connDb.Open();
string comandoSql =
"SELECT * FROM Prodotti WHERE idProdotto = " + idPr + "";
command.CommandText = comandoSql;
command.Connection = connDb;
rsProdotto = command.ExecuteReader();
if (rsProdotto.Read())
{
    lblDescrizione.Text = rsProdotto["descrizione"].ToString();
    lblPrezzo.Text = rsProdotto["Prezzo"].ToString();
    lblQta.Text = rsProdotto["Qta"].ToString();
}
else
    lblErrore.Text = "codice prodotto inesistente";
rsProdotto.Close();
}
}

```

IL RISULTATO

VISUALIZZA PRODOTTO

CODICE PRODOTTO: gtx1

Descrizione: scheda grafica

Prezzo: 400

Quantità: 5

CERCA

fig. 6 Caso 1: Il prodotto richiesto è presente

VISUALIZZA PRODOTTO

CODICE PRODOTTO: xyz

Descrizione:

Prezzo:

Quantità:

CERCA

codice prodotto inesistente

fig. 7 Caso 2: Il prodotto richiesto non è presente

LABORATORIO

IL PROBLEMA Visualizza i dati di tutti i prodotti ordinati per codice prodotto.

L'ANALISI Per risolvere l'esercizio facciamo riferimento a GridView. Si tratta di un controllo ASP.NET che permette la visualizzazione di dati in formato tabellare ed è l'ideale per visualizzare le tabelle del database. Dopo aver stabilito la connessione con il database facendo uso di un oggetto OleDbConnection, fai eseguire una query utilizzando il comando ExecuteReader: mettendo i risultati in un oggetto OleDbDataReader (rsProdotti), mediante un ciclo ogni riga di rsProdotti è aggiunta nella tabella dt di tipo DataTable. Per visualizzare i dati utilizza una GridView con la DataTable come origine dati. Il comando DataBind associa i dati della tabella alla GridView.

L'INTERFACCIA

Elenco Prodotti			
IdProdotto	Descrizione	Prezzo	Quantità
Associato a dati	Associato a dati	Associato a dati	Associato a dati
Associato a dati	Associato a dati	Associato a dati	Associato a dati
Associato a dati	Associato a dati	Associato a dati	Associato a dati
Associato a dati	Associato a dati	Associato a dati	Associato a dati
Associato a dati	Associato a dati	Associato a dati	Associato a dati

CODIFICA C#.NET (DATABASE ACCESS)

```
using System;
using System.Data;
using System.Data.OleDb;
public partial class ElencoProdotti : System.Web.UI.Page
{
    protected void Btn_Click(object sender, EventArgs e)
    {
        OleDbConnection connDb = new OleDbConnection();
        OleDbCommand command = new OleDbCommand();
        OleDbDataReader rsProdotto;
        DataTable dt = new DataTable();
        connDb.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;" +
            "Data Source=|DataDirectory|/magazzino.accdb";
        connDb.Open();
        string query = "SELECT * FROM Prodotti ";
        command.Connection = connDb;
        command.CommandText = query;
        rsProdotto = command.ExecuteReader();
        if (rsProdotto.HasRows)
        {
            dt.Columns.Add("idProdotto");
            dt.Columns.Add("descrizione");
            dt.Columns.Add("prezzo");
            dt.Columns.Add("qta");
            while (rsProdotto.Read())
            {
                DataRow row = dt.NewRow();
```

```

        row["idProdotto"] = rsProdotto["idProdotto"].ToString();
        row["descrizione"] = rsProdotto["descrizione"].ToString();
        row["prezzo"] = rsProdotto["prezzo"].ToString();
        row["qta"] = rsProdotto["qta"].ToString();
        dt.Rows.Add(row);
    }
    GrdProdotti.DataSource = dt;
    GrdProdotti.DataBind();
    rsProdotto.Close();
    connDb.Close();

}
}
}

```

CODIFICA C#.NET (DATABASE SQL SERVER)

```

using System;
using System.Data;
using System.Data.SqlClient;

public partial class Elenco : System.Web.UI.Page
{
    protected void Btn_Click(object sender, EventArgs e)
    {
        SqlConnection connDb = new SqlConnection();
        SqlCommand command = new SqlCommand();
        SqlDataReader rsProdotto;
        DataTable dt = new DataTable();

        connDb.ConnectionString = "Data Source = (LocalDB)\MSSQLLocalDB; " +
        " AttachDbFileName =|DataDirectory|magazzino.mdf; " +
        " Initial Catalog = magazzino.mdf";
        connDb.Open();
        string comandoSql = "SELECT * FROM Prodotti";
        command.CommandText = comandoSql;
        command.Connection = connDb;
        rsProdotto = command.ExecuteReader();
        if (rsProdotto.HasRows)
        {
            dt.Columns.Add("idProdotto");
            dt.Columns.Add("descrizione");
            dt.Columns.Add("prezzo");
            dt.Columns.Add("qta");

            while (rsProdotto.Read())
            {
                DataRow row = dt.NewRow();
                row["idProdotto"] = rsProdotto["idProdotto"].ToString();
                row["descrizione"] = rsProdotto["descrizione"].ToString();
                row["prezzo"] = rsProdotto["prezzo"].ToString();
                row["qta"] = rsProdotto["qta"].ToString();
            }
        }
    }
}

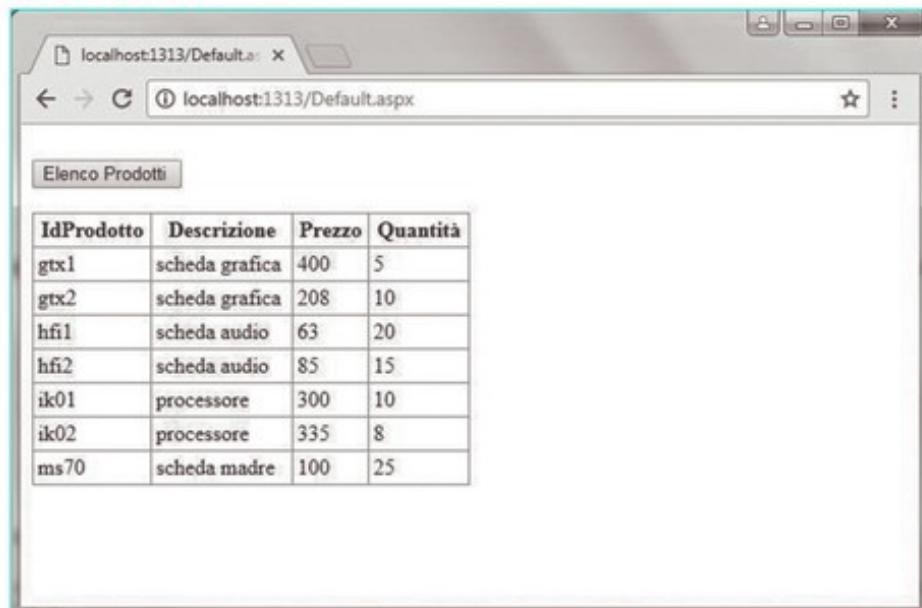
```

```

        dt.Rows.Add(row);
    }
    GrdProdotti.DataSource = dt;
    GrdProdotti.DataBind();
    rsProdotto.Close();
    connDb.Close();
}
}
}
}

```

L'ESECUZIONE



FISSA LE CONOSCENZE

1. Come si apre una connessione con un database?
2. Quale comando si usa per scrivere una query SQL?
3. Che cosa contiene un'istanza dell'oggetto DataReader?
4. Come si controlla se il risultato di una query ha prodotto almeno una riga?
5. Come si identifica un campo del DataReader?
6. Che cosa restituisce il metodo Read()?
7. Come si può controllare l'esito di una ricerca per chiave primaria?
8. Come si scandiscono tutte le righe di un DataReader?

5. INSERIMENTO E MODIFICA DATI

La modifica dei dati avviene utilizzando il metodo ExecuteNonQuery delle classi OleDbCommand e SqlCommand, che permette di eseguire sia le istruzioni SQL per la modifica dei dati (INSERT, DELETE, UPDATE) che l'istruzione per la definizione di una tabella (CREATE TABLE).

Il metodo ExecuteNonQuery restituisce inoltre un numero, che rappresenta il numero di record modificati (o inseriti o cancellati); restituisce – 1 in caso di fallimento.

È possibile ottenere tale numero assegnando a una variabile di tipo intero il risultato di ExecuteNonQuery():

```
int NumRighe = comandoSql.ExecuteNonQuery();
```

LABORATORIO

IL PROBLEMA Inserisci nella tabella Prodotti i dati di un nuovo prodotto.

L'ANALISI Per inserire un nuovo prodotto, prendi in input i suoi dati, apri la connessione al database, prepara il comando INSERT di SQL e, attraverso il metodo ExecuteNonQuery, fai eseguire il comando. Il controllo che il codice del prodotto non sia già presente avviene nel costrutto Try/Catch.

L'INTERFACCIA

Inserimento Prodotto

Codice
Descrizione
Prezzo
Quantità

Inserisci Annulla

LblMessaggio

CODIFICA C#.NET (DATABASE ACCESS)

```
using System;
using System.Data.OleDb;

public partial class Inserisci : System.Web.UI.Page
{
    protected void BtnInserisci_Click(object sender, EventArgs e)
    {
        OleDbConnection connDb = new OleDbConnection();
        OleDbCommand command = new OleDbCommand();
        String idpr = TxtIdProdotto.Text;
        String desc = TxtDescrizione.Text;
        float prezzo = float.Parse(TxtPrezzo.Text);
        int qta = int.Parse(TxtQta.Text);
        connDb.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;" +
            "Data Source=|DataDirectory|/magazzino.accdb";
        connDb.Open();
        string comando = "insert into Prodotti " +
```

```

    " (IdProdotto,Descrizione,Prezzo,Qta )" +
    " values (" + idpr + "," + desc + "," + prezzo + "," + qta + ")";
command.Connection = connDb;
command.CommandText = comando;
try
{ //Modifica/cancella/aggiunge nella tabella del DB
    command.ExecuteNonQuery();
    LblMessaggio.Text = "Prodotto inserito correttamente ";
}
catch
{
    LblMessaggio.Text = "errore nell'inserimento ";
}
connDb.Close();
}
}

```

CODIFICA C#.NET (DATABASE SQL SERVER)

```

using System;
using System.Data.SqlClient;

public partial class Inserisci : System.Web.UI.Page
{
    protected void BtnInserisci_Click(object sender, EventArgs e)
    {
        SqlConnection connDb = new SqlConnection();
        SqlCommand command = new SqlCommand();
        String idpr = TxtIdProdotto.Text;
        String desc = TxtDescrizione.Text;
        float prezzo = float.Parse(TxtPrezzo.Text);
        int qta = int.Parse(TxtQta.Text);
        ConnessioneDb.ConnectionString = "Data Source = "+
        "(LocalDB)\MSSQLLocalDB; AttachDbFileName =" +
        "|DataDirectory|magazzino.mdf; Initial Catalog = magazzino.mdf";
        connDb.Open();
        string comandoSql = "insert into Prodotti " +
            " (IdProdotto,Descrizione,Prezzo,Qta )" +
            " values (" + idpr + "," + desc + "," + prezzo + "," + qta + ")";
        command.CommandText = comandoSql;
        command.Connection = connDb;
        try
        { //Modifica/cancella/aggiunge nella tabella del DB
            command.ExecuteNonQuery();
            LblMessaggio.Text = "Prodotto inserito correttamente" ;
        }
        catch
        {
            LblMessaggio.Text = "errore nell'inserimento ";
        }
        connDb.Close();
    }
}

```

IL RISULTATO

The screenshot shows a web page titled "Inserimento Prodotto". It contains four input fields: "Codice" with value "xxg2", "Descrizione" with value "tastiera", "Prezzo" with value "35", and "Quantità" with value "10". Below the form are two buttons: "Inserisci" and "Annulla". A message at the bottom of the page reads "Prodotto inserito correttamente" (Product inserted correctly).

fig. 8 Caso 1: Inserimento corretto

The screenshot shows a web page titled "Inserimento Prodotto". It contains four input fields: "Codice" with value "gtx1", "Descrizione" with value "tastiera", "Prezzo" with value "15", and "Quantità" with value "5". Below the form are two buttons: "Inserisci" and "Annulla". A message at the bottom of the page reads "errore nell'inserimento" (insertion error).

fig. 9 Caso 2: Inserimento errato

Per le operazioni di modifica e cancellazione di un prodotto si utilizza la stessa modalità presentata per l'inserimento, con la sola modifica dell'istruzione SQL nella quale si inseriscono i comandi UPDATE o DELETE in funzione della richiesta.

FISSA LE CONOSCENZE

1. Quando si usa il metodo ExecuteNonQuery?
2. Che cosa restituisce il metodo ExecuteNonQuery?
3. Quale comando si usa per estrarre i dati da una tabella?
4. Quale comando SQL è necessario per modificare il contenuto di una riga di una tabella?
5. Quale comando SQL è necessario per eliminare righe dal database?

6. LOGIN

In questa Lezione è illustrato come controllare l'esistenza di dati all'interno di un database, in particolare come si gestisce un login di connessione.

LABORATORIO

IL PROBLEMA Costruisci una pagina dinamica per effettuare il login alla videoteca.

L'ANALISI I valori della Userid e della password dell'utente vanno inseriti nella pagina HTML di interfaccia. Quando l'utente clicca sul pulsante Accedi, viene attivato il codice in cui è impostata la query SQL sulla tabella Utenti. Questa verifica che l'utente sia effettivamente già registrato nel database e che la password inserita nel form corrisponda a quella memorizzata nel database.

L'INTERFACCIA

Lasciamo allo studente il compito di ricavare la codifica della pagina HTML dall'interfaccia proposta.

Codifica VB.NET

The screenshot shows a Windows application window titled "LOGIN". Inside the window, there are two text input fields: one labeled "Userid" and another labeled "Password". Below these fields are two buttons: "Annulla" (Cancel) on the left and "Accedi" (Login) on the right. At the bottom of the window, there is a label control with the text "LblMessaggio".

CODIFICA C#.NET (DATABASE ACCESS)

```
using System;
using System.Data.OleDb;

public partial class Inserisci : System.Web.UI.Page
{
    protected void btnLogin_Click(object sender, EventArgs e)
    {
        OleDbConnection connection = new OleDbConnection();
        OleDbCommand command = new OleDbCommand();
        OleDbDataReader rsUtente;
        LblMessaggio.Text = " ";
        string user = TxtUser.Text;
        string pw = TxtPsw.Text;
        connDb.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;" +
            "Data Source=|DataDirectory|/magazzino.accdb";
        connection.Open();
        string query = "SELECT * FROM Utenti where userId = '" + user + "' " +
            "and psw = '" + pw + "'";
        command.Connection = connection;
        command.CommandText = query;
        rsUtente = command.ExecuteReader();
        if (rsUtente.Read())
        {
            LblMessaggio.Text = "Benvenuto signor " + rsUtente["nome"] + " " + rsUtente["cognome"];
        }
        else
```

```

    {
        LblMessaggio.Text = "username o password errate";
    }
    rsUtente.Close();
    connection.Close();
}

protected void btnAnnulla_Click(object sender, EventArgs e)
{
    TxtUser.Text = "";
    TxtPsw.Text = "";
    LblMessaggio.Text = "";
}
}

```

CODIFICA C#.NET (DATABASE SQL SERVER)

```

using System;
using System.Data.OleDb;

public partial class Inserisci : System.Web.UI.Page
{
    protected void btnLogin_Click(object sender, EventArgs e)
    {
        LblMessaggio.Text = " ";
        string user = TxtUser.Text;
        string pw = TxtPsw.Text;
        SqlConnection connection = new SqlConnection();
        connection.ConnectionString = "Data Source = (LocalDB)\MSSQLLocalDB; " +
            " AttachDbFileName =|DataDirectory|magazzino.mdf; Initial Catalog = magazzino.mdf";
        connection.Open();
        string sq = "SELECT * FROM Utenti where userId = '" + user + "'+" +
            " and psw = '" + pw + "'";
        SqlCommand command = new SqlCommand();
        command.Connection = connection;
        command.CommandText = sq;
        SqlDataReader rsUtente;
        rsUtente = command.ExecuteReader();
        command.Dispose();
        if (rsUtente.Read())
            LblMessaggio.Text = "Benvenuto signor " + rsUtente["nome"] + " " + rsUtente["cognome"];
        else
        {
            LblMessaggio.Text = "username o password errate";
        }
        rsUtente.Close();
        connection.Close();
    }

    protected void btnAnnulla_Click(object sender, EventArgs e)
    {
        TxtUser.Text = "";
    }
}

```

```
        TxtPsw.Text = "";
        LblMessaggio.Text = "";
    }
}
```

L'ESECUZIONE

The screenshot shows a login form titled "LOGIN". It has two input fields: "UserId" containing "utente1" and "Password" which is empty. Below the form are two buttons: "Annulla" and "Accedi". Underneath the form, the text "Benvenuto signor utente1" is displayed.

fig. 10 Caso 1: Userid e password corrette

The screenshot shows a login form titled "LOGIN". It has two input fields: "UserId" containing "utente1" and "Password" which is empty. Below the form are two buttons: "Annulla" and "Accedi". Underneath the form, the text "Password errata" is displayed.

fig. 11 Caso 2: Userid corretto a password errata

The screenshot shows a login form titled "LOGIN". It has two input fields: "UserId" containing "utente" and "Password" which is empty. Below the form are two buttons: "Annulla" and "Accedi". Underneath the form, the text "User inesistente" is displayed.

fig. 12 Caso 3: Userid errata

FISSA LE CONOSCENZE

1. Quali controlli sono necessari per effettuare il login?
2. Di che tipo è l'oggetto dove è inserita la password?
3. Per accedere ai siti è sempre necessario effettuare il login?
Quando l'accesso può essere libero?

7. IMPORTARE ED ESPORTARE DATI

Quando si deve popolare un database aziendale, la tecnica illustrata nelle precedenti lezioni può risultare poco efficiente, poiché prevede di inserire da tastiera i dati per ciascun record.

Se la cardinalità della tabella è elevata, l'operazione manuale può richiedere tempo e comportare errori di digitazione. Nella realtà possono presentarsi diversi scenari e, per ciascuno di essi, si può adottare la soluzione adeguata. Per esempio, i dati potrebbero essere memorizzati in una cartella Excel o contenuti in un file di testo, si potrebbe avere a disposizione Access oppure no.

Quando non è possibile utilizzare Access per compiere queste operazioni, si può risolvere il problema scrivendo un apposito programma, come è indicato nel prossimo esercizio guidato.

LABORATORIO

IL PROBLEMA Inserisci nella base di dati le informazioni relative a nuovi prodotti contenute in un file di testo.

L'ANALISI Dopo aver effettuato la connessione con la base di dati, viene eseguito un ciclo di lettura sul file di testo: per ogni riga si separano i campi con il metodo Split, che restituisce in un vettore (Dati) i singoli dati. Può essere così preparato il comando SQL INSERT per inserire il nuovo prodotto nella tabella. Si lascia allo studente la definizione del programma che realizza l'esportazione dei dati dalla tabella Prodotti a un file di testo.

CODIFICA C#.NET (DATABASE ACCESS)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Data.OleDb;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string nomeDB= "magazzino.accdb";
        OleDbConnection connDB = new OleDbConnection();
        OleDbCommand istruzioneSQL = new OleDbCommand();
        string riga;
        connDB.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;" +
        "Data Source=|DataDirectory|/magazzino.accdb";
```

```

connDB.Open();
string[] dati;
StreamReader file1 = new StreamReader(Server.MapPath("~/miofile.txt"));
while (!file1.EndOfStream)
{
    riga = file1.ReadLine();
    dati = riga.Split(',');
    string idpr = dati[0];
    string desc = dati[1];
    float prezzo = float.Parse(dati[2]);
    int qta = int.Parse(dati[3]);
    istruzioneSQL.CommandText = "Insert into Prodotti " +
        " (IdProdotto, Descrizione, Prezzo, Qta )" +
        " values (" + idpr + "," + desc + "," + prezzo + "," + qta + ")";
    istruzioneSQL.Connection = connDB;
    try
    { //Modifica/cancella/aggiunge nella tabella del DB
        istruzioneSQL.ExecuteNonQuery();
    }
    catch (Exception objError )
    {
        lblMessaggio.Text = "errore nell'inserimento" + objError.Message;
    }
}
connDB.Close();
file1.Close();
}
}

```

CODIFICA C#.NET (DATABASE SQL SERVER)

```

using System;
using System.IO;
using System.Data.SqlClient;

public partial class ImportaDaFile : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        SqlConnection connDb = new SqlConnection();
        SqlCommand istruzioneSQL = new SqlCommand();
        string riga;
        connDb.ConnectionString = "Data Source = (LocalDB)\MSSQLLocalDB; " +
            " AttachDbFileName =|DataDirectory|magazzino.mdf; " +
            "Initial Catalog = magazzino.mdf";
        connDb.Open();
        string[] dati;
        StreamReader file1 = new StreamReader(Server.MapPath("~/miofile.txt"));
        while (!file1.EndOfStream)
        {

```

```

riga = file1.ReadLine();
dati = riga.Split(',');
string idpr = dati[0];
string desc = dati[1];
float prezzo = float.Parse(dati[2]);
int qta = int.Parse(dati[3]);
istruzioneSQL.CommandText = "Insert into Prodotti " +
    " values (" + idpr + "," + desc + "," + prezzo + "," + qta + ")";
istruzioneSQL.Connection = connDb;
try
{ //Modifica/cancella/aggiunge nella tabella del DB
    istruzioneSQL.ExecuteNonQuery();
}
catch (Exception objError)
{
    lblMessaggio.Text = "errore nell'inserimento" + objError.Message;
}
}

connDb.Close();
file1.Close();
}
}

```

Lavorando con Access, le operazioni per importare ed esportare dati risultano piuttosto semplici:

- nella scheda Dati esterni ci sono i comandi che permettono di riutilizzare dati da numerose altre fonti;
- nel gruppo Importa sono visualizzate le icone corrispondenti ai formati di dati supportati per l'importazione e il collegamento;
- nel gruppo Esporta sono visualizzate le icone corrispondenti ai formati supportati per l'esportazione da Access in altri formati.

In ogni gruppo è inoltre possibile fare clic su Altro, per visualizzare altri formati utilizzabili.

Cliccando sull'icona corrispondente nella barra in alto, è possibile importare dati da un file ([fig. 13](#)) e anche, viceversa, esportarli nel formato Excel ([fig. 14](#)).

[fig. 13 Importare dati.](#)





Nel caso dell'importazione da file (fig. 15), si apre una finestra nella quale va indicato il percorso della cartella da importare e la modalità di importazione, tenendo presente che sul foglio Excel la prima riga deve contenere le intestazioni delle colonne (fig. 16).

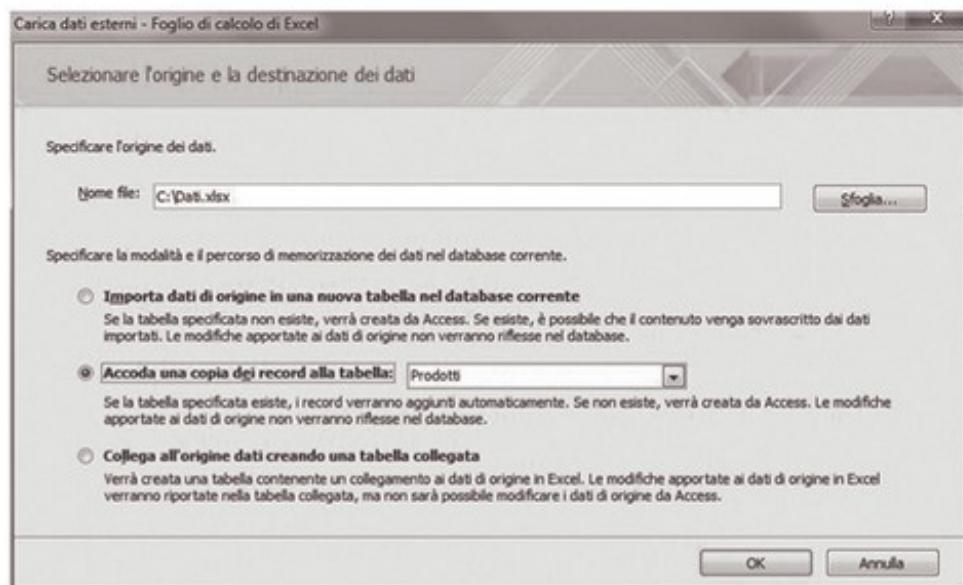


fig. 14 Esportare dati.

fig. 15 Scelta del file da importare.

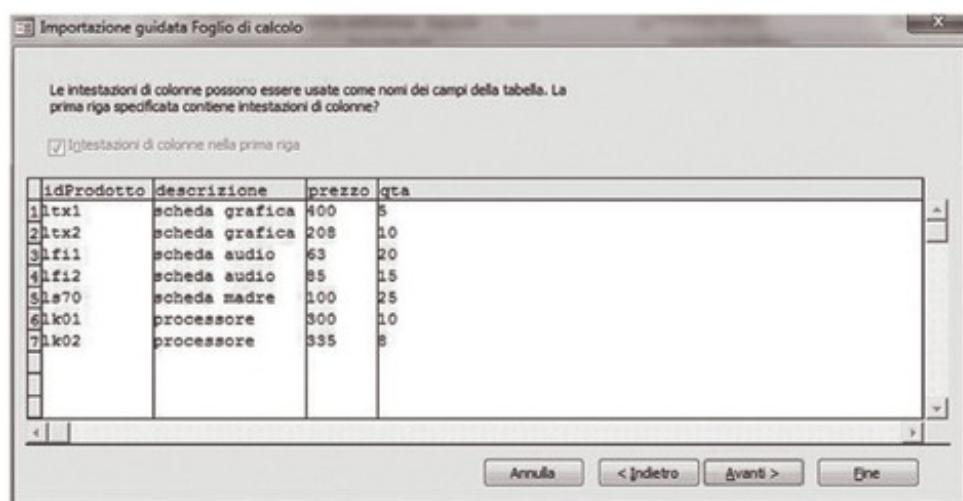
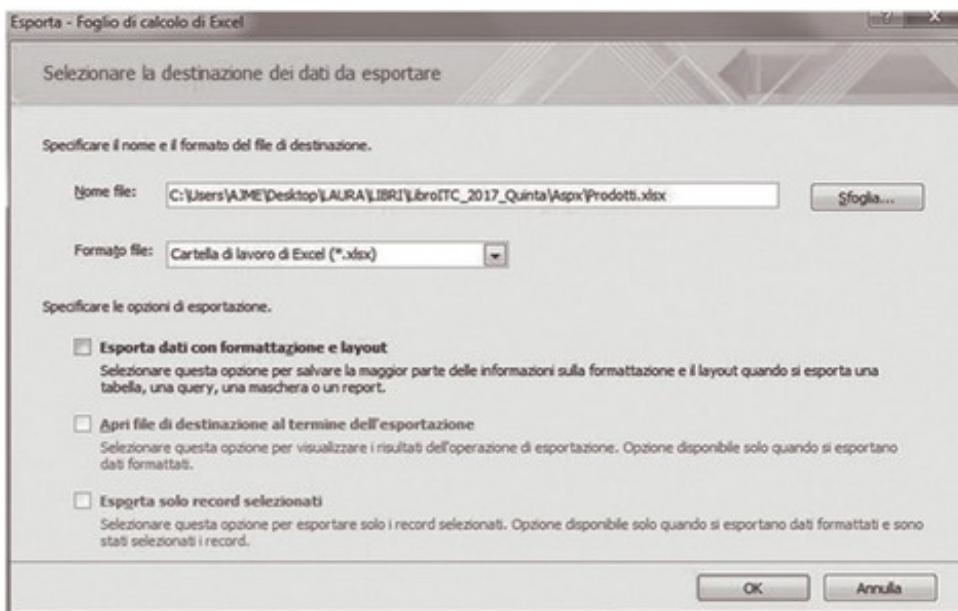


fig. 16 Impostazioni per la tabella

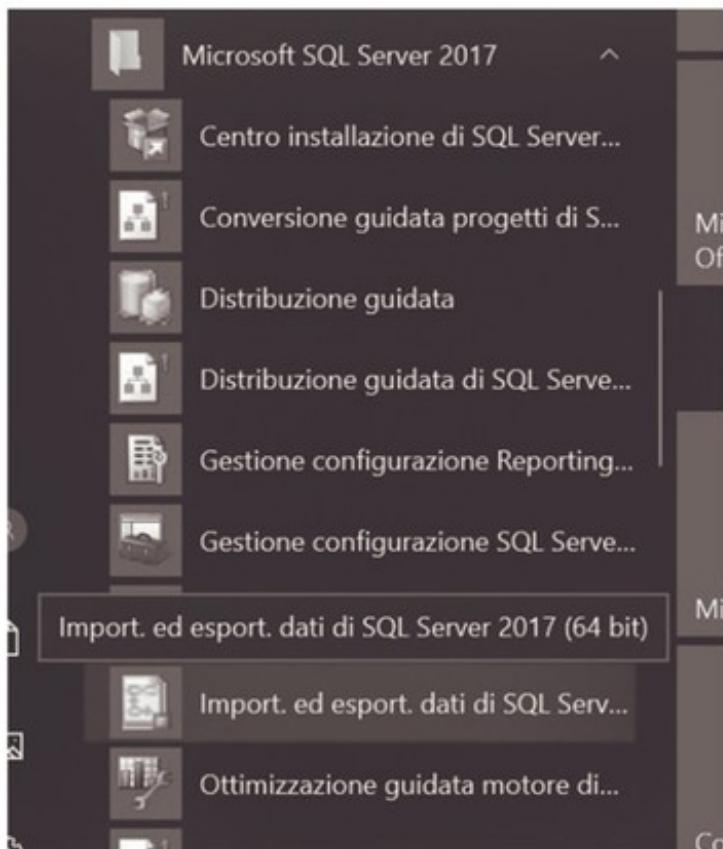
Per esportare i dati la procedura è altrettanto semplice: dopo aver cliccato sull'icona del formato scelto per l'esportazione nella scheda Esporta, è sufficiente indicare il nome della cartella che conterrà i dati esportati (fig. 17).

fig. 17 Scegliere il formato del file da esportare.



Quando si utilizza un database di SQL Server è utile avere installata SQL Server Importa/Esporta, che consente di copiare facilmente i dati da un'origine a una diversa destinazione (**fig. 18**)

fig. 18 SQL Server Importa/Esporta è installato



Per esportare i dati, avviare l'Importazione/Esportazione guidata SQL Server e nelle pagine della procedura guidata corrispondenti selezionare l'origine e la destinazione dei dati. Nel caso preso in esame la destinazione è un file con estensione csv: (**fig. 19** e **fig. 20**).

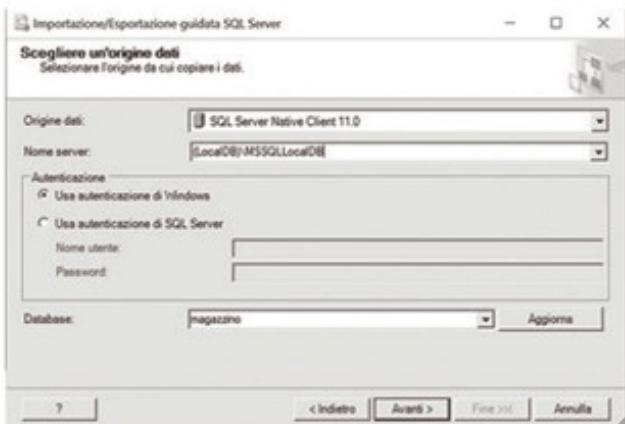


fig. 19 Scelta dell'origine dati

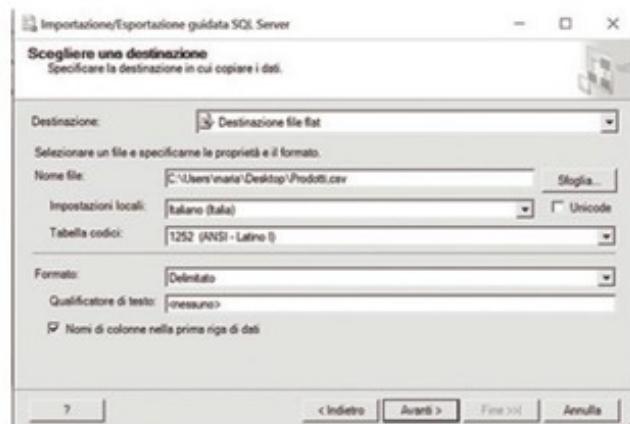


fig. 20 Impostazioni del file origine dati

Successivamente bisogna selezionare se copiare i dati da tabelle o viste, oppure copiare il risultato da una query. (fig. 21)

Bisogna poi specificare il delimitatore di riga e di colonna da utilizzare nel file di destinazione. Infine salvare ed eseguire un pacchetto (fig. 22).

Il procedimento di esportazione è analogo al precedente e si effettua invertendo l'origine e la destinazione dei dati.

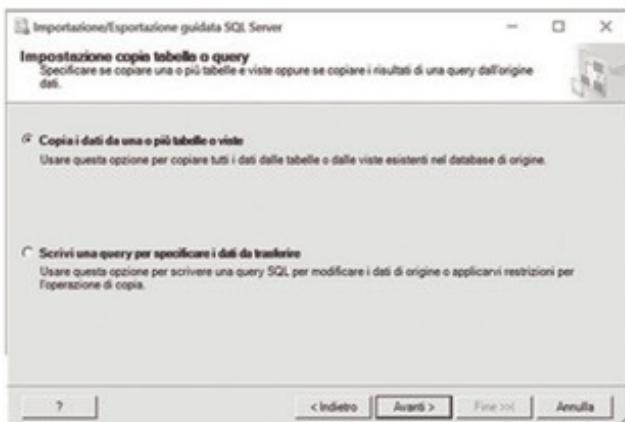


fig. 21 Impostazione della tabella



fig. 22 Impostazioni per copia da tabella o da query

FISSA LE CONOSCENZE

1. Quando l'inserimento dei dati da un form non è conveniente?
2. Come è strutturato il programma che esporta i dati da una tabella a un file di testo?
3. Come sono trattate le intestazioni delle colonne quando si importano dati da un foglio Excel?
4. Quali sono i passi per importare un foglio Excel in Access?
5. Come è strutturato il programma che importa i dati da un file di testo in una tabella?

8. ESERCIZIO COMPLETO IN ASP.NET

LABORATORIO

IL PROBLEMA Si vuole realizzare un sito web per la gestione online di un museo, in particolare si vogliono implementare le seguenti richieste: indicare il numero di biglietti emessi per una determinata esposizione; calcolare il ricavato della vendita dei biglietti di una data esposizione.

Il database di riferimento è il seguente:

Visita (id_visita, titolo, tipo, tariffa, dataInizio, dataFine);

Biglietto (id_biglietto, dataValidità, cod_visita, cod_categoria);

Servizio (id_servizio, descrizione, prezzo);

Categoria (id_categoria, descrizione, tipoDocumento, percentualeSconto);

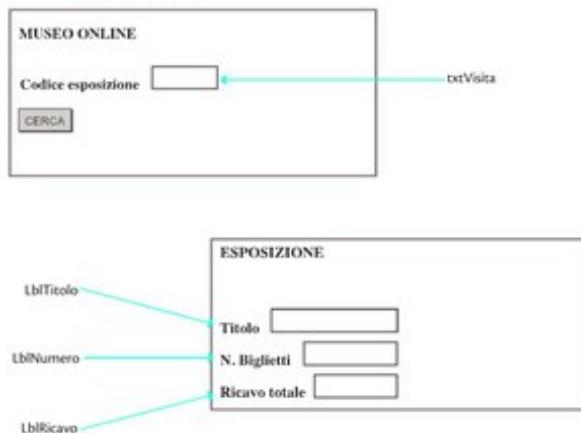
Associare (id_biglietto, id_servizio).

L'ANALISI Dobbiamo progettare il layout del sito per la gestione del museo e quindi realizzare i programmi che si interfaceranno con il database.

La prima operazione da fare è disegnare il layout delle pagine che compongono il sito web.

Nella Home page HTML (*esposizione.html*), sarà presente una casella di testo in cui inserire il codice dell'esposizione di cui cercare le informazioni e un tasto CERCA, per richiamare la pagina successiva che conterrà i dati relativi all'esposizione e i risultati delle query necessarie per indicare il numero di biglietti emessi e il ricavato. Il tasto CERCA richiama il programma *esposizione.php*.

L'INTERFACCIA



CODIFICA C#.NET (DATABASE ACCESS)

```
using System;
using System.Data.OleDb;
public partial class _Default : System.Web.UI.Page
{
    protected void BtnCerca_Click(object sender, EventArgs e)
    {
        OleDbConnection connDb = new OleDbConnection();
```

```

OleDbCommand comandoSql = new OleDbCommand();
OleDbDataReader rsVisita;
String idVisita;
String titolo;
int tariffa;
connDb.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;" +
    "Data Source=|DataDirectory|museo.accdb";
connDb.Open();
LblTitolo.Text = " ";
LblNumero.Text = " ";
LblTitolo.Text = " ";
LblRicavo.Text = " ";
try
{
    idVisita = txtVisita.Text;
    string query= "Select titolo,tariffa From Visita" +
        " Where id_Visita = '" + idVisita + "'";
    comandoSql.CommandText = query;
    comandoSql.Connection = connDb;
    rsVisita = comandoSql.ExecuteReader();
    if (!rsVisita.Read())
    {
        LblNumero.Text = "codice esposizione inesistente";
        rsVisita.Close();
        connDb.Close();
    }
    else
    {
        titolo = rsVisita.GetString(0);
        tariffa = rsVisita.GetInt32(1);
        rsVisita.Close();
        comandoSql.Dispose();
        query = "Select Count(*) as numeroBiglietti From Biglietto " +
            "Where cod_visita = '" + idVisita + "'";
        comandoSql.CommandText = query;
        comandoSql.Connection = connDb;
        rsVisita = comandoSql.ExecuteReader();
        rsVisita.Read();
        LblTitolo.Text = " Titolo: " + titolo;
        int n = rsVisita.GetInt32(0);
        LblNumero.Text = " Numero biglietti: " + n.ToString();
        rsVisita.Close();
        comandoSql.Dispose();
        query = "Select Sum(tariffa-(tariffa * percentualeSconto/100))" +
            " as totale From Visita,Biglietto, Categoria " +
            " Where id_visita = cod_visita " +
            " and cod_categoria = id_categoria" +
            " And cod_visita = '" + idVisita + "'";
        comandoSql.CommandText = query;
        comandoSql.Connection = connDb;
        rsVisita = comandoSql.ExecuteReader();
        rsVisita.Read();
    }
}

```

```

        double n1 = rsVisita.GetDouble(0);
        lblRicavo.Text = " Ricavo totale: " + n1.ToString();
    }
    rsVisita.Close();
    connDb.Close();
}
catch
{
    Response.Write("Errore ");
}
}
}

```

CODIFICA C#.NET (DATABASE SQL SERVER)

```

using System;
using System.Data.SqlClient;
public partial class _Default : System.Web.UI.Page
{
    protected void BtnCerca_Click(object sender, EventArgs e)
    {
        SqlConnection connDb = new SqlConnection();
        SqlCommand comandoSql = new SqlCommand();
        SqlDataReader rsVisita;
        String idVisita;
        String titolo;
        //float tariffa;
        connDb.ConnectionString = "Data Source = (LocalDB)\MSSQLLocalDB; " +
            "AttachDbFileName =|DataDirectory|museo.mdf; " +
            "Initial Catalog = magazzino.mdf";
        connDb.Open();
        LblTitolo.Text = " ";
        LblNumero.Text = " ";
        LblTitolo.Text = " ";
        LblRicavo.Text = " ";
        try
        {
            idVisita = txtVisita.Text;
            string query = "Select titolo,tariffa From Visita" +
                " Where id_Visita = " + idVisita + "";
            comandoSql.CommandText = query;
            comandoSql.Connection = connDb;
            rsVisita = comandoSql.ExecuteReader();
            if (!rsVisita.Read())
            {
                LblNumero.Text = "codice esposizione inesistente";
                rsVisita.Close();
                connDb.Close();
            }
            else
            {
                titolo = rsVisita.GetString(0);
                double tariffa = rsVisita.GetDouble(1);
                rsVisita.Close();
            }
        }
    }
}

```

```

        comandoSql.Dispose();
        query = "Select Count(*) as numeroBiglietti From Biglietto " +
            "Where cod_visita = " + idVisita + "";
        comandoSql.CommandText = query;
        comandoSql.Connection = connDb;
        rsVisita = comandoSql.ExecuteReader();
        rsVisita.Read();
        LblTitolo.Text = " Titolo: " + titolo;
        int n = rsVisita.GetInt32(0);
        LblNumero.Text = " Numero biglietti: " + n.ToString();
        rsVisita.Close();
        comandoSql.Dispose();
        query = "SELECT Sum(tariffa - (tariffa * percentualeSconto / 100))" +
            " as totale From Visita,Biglietto, Categoria " +
            " Where id_visita = cod_visita " +
            " And cod_categoria = id_categoria" +
            " And cod_visita = " + idVisita + "";
        comandoSql.CommandText = query;
        comandoSql.Connection = connDb;
        rsVisita = comandoSql.ExecuteReader();
        rsVisita.Read();
        double n1 = rsVisita.GetDouble(0);
        lblRicavo.Text = " Ricavo totale: " + n1.ToString();
    }
    rsVisita.Close();
    connDb.Close();
}
catch
{
    Response.Write("Errore ");
}
}

```





RIPASSIAMO INSIEME

HTML e ASP.NET

Attraverso un **form HTML** il client è in grado di inviare dei dati al server; nel form vi sono elementi in cui l'utente inserisce dati.

Nella Lezione si esamina l'**invio** e il **recupero di dati** a seconda del tipo di elemento presente nel form HTML.

Passaggio dei parametri

Da una pagina ASP.NET è possibile richiamarne un'altra. Le pagine possono scambiarsi informazioni tramite un **cookie**, una **stringa di query** o con le **variabili di sessione**.

Connessione al Db

L'accesso al database avviene utilizzando la **tecnologia ADO.NET**, che utilizza gli oggetti della libreria System.Data.OleDb se si lavora con un database Access. Se invece il servizio è fornito dal **DBMS SQL Server**, è necessario includere la libreria System.Data.SqlClient.

Visualizzazione dati

Il risultato di una query è memorizzato in **DataReader**. Il metodo Read() dell'oggetto DataReader viene usato per ottenere una riga dai risultati della query.

Se la query restituisce più righe di risultato, per visualizzarle tutte è necessario eseguire un ciclo di lettura sul DataReader.

Inserimento e modifica dati

L'inserimento dei dati all'interno di una tabella di database (così come la cancellazione e la modifica) avviene utilizzando il comando **ExecuteNonQuery**.

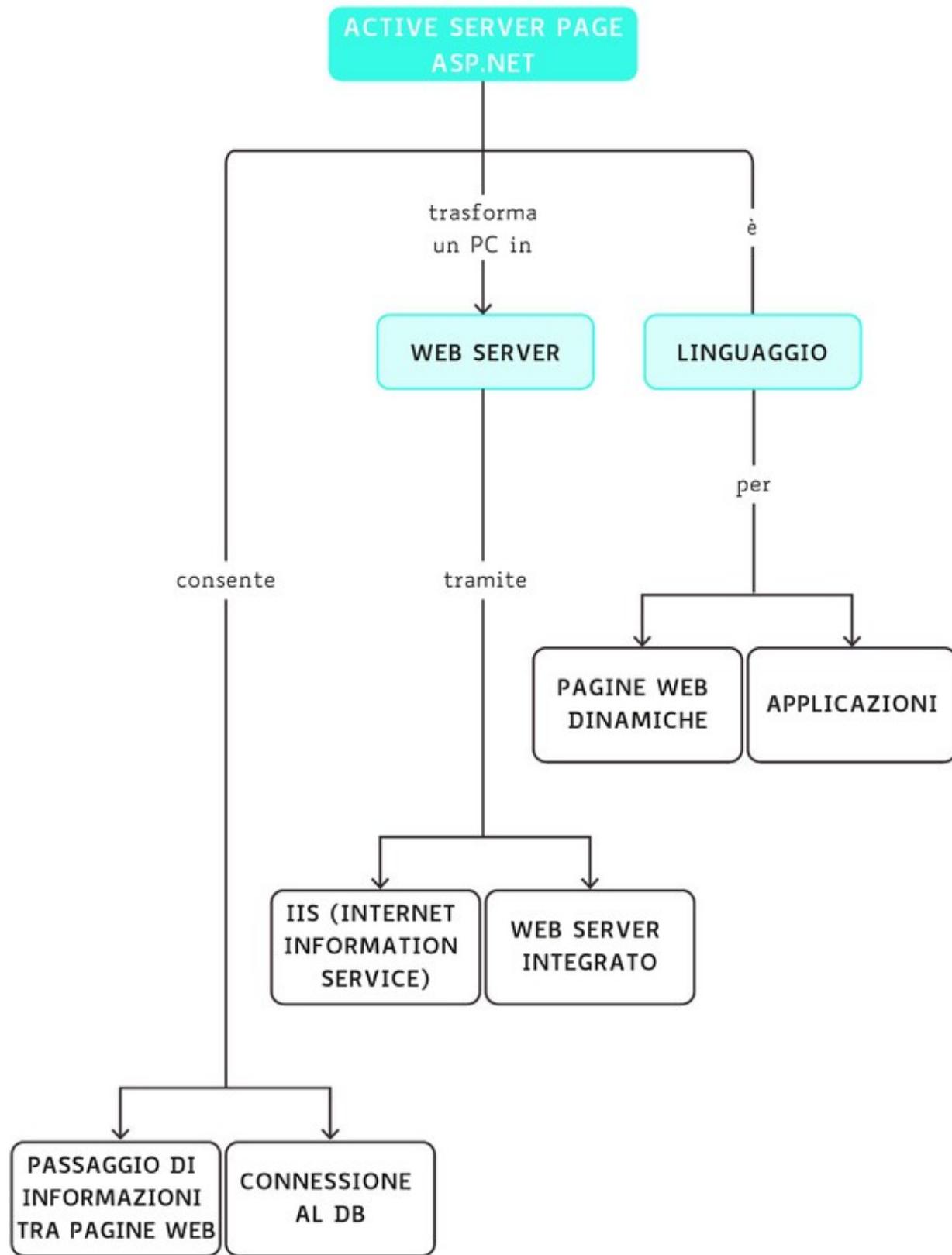
Login

È possibile **controllare l'accesso** a un database da parte di un utente. I messaggi di errore compaiono quando viene inserita la Userid errata o quando la Userid è corretta, ma non lo è la password.

Importare ed esportare dati

Per inserire o per estrarre i dati da un database, si possono sfruttare le funzioni di **Access**. Esse consentono agevolmente di importare/esportare dati provenienti da altre fonti (Excel, file di testo). Se non si ha a disposizione Access, si possono importare i dati registrati su un file di testo, scrivendo un **apposito programma** che legge i dati dal file e li inserisce nel database con il comando SQL INSERT.





TEST



TEST

Svolgi il test interattivo

Vero o falso?

1. ASP.NET crea pagine dinamiche. V F
2. Un programma ASP.NET è eseguito dal client. V F
3. L'output di una pagina dinamica è una pagina HTML. V F
4. La durata di una variabile di sessione non è modificabile. V F
5. Il cookie risiede sul server. V F
6. Per visualizzare il risultato di una query, si usa sempre un ciclo. V F
7. Per cancellare i dati da una tabella, si usa il metodo ExecuteNonQuery. V F
8. Il PostBack è eseguito quando una pagina richiama se stessa. V F

Scelta multipla (una sola è la risposta esatta)

9. Una pagina ASP.NET è formata da:
 - A codice HTML.
 - B codice HTML e codice C#.NET.
 - C codice C#.NET.
 - D richiami al database ADO.NET.
10. L'istruzione Response.Redirect serve per:
 - A richiedere l'esecuzione di una pagina sul client.
 - B richiedere l'esecuzione di una pagina sul server.
 - C richiamare una pagina ASP.NET.
 - D ridirigere l'output.
11. Il metodo ExecuteNonQuery non può essere usato per:
 - A eseguire un comando SQL di inserimento.
 - B eseguire una query SQL.
 - C eseguire un comando SQL di modifica.
 - D eseguire un comando SQL di cancellazione.

PREPARATI PER IL COLLOQUIO ORALE

1. Come è strutturata una pagina ASP.NET? [vedi lez. 1]
2. A che cosa servono i cookie e come vengono utilizzati? [vedi lez. 2]
3. In quali casi e come vengono utilizzati i parametri? [vedi lez. 2]
4. Su quale tecnologia è basata ASP.NET per la gestione di database? [vedi lez. 3]
5. Come viene esaminato il risultato del comando ExecuteReader()? [vedi lez. 4]
6. Come avviene una query di cancellazione? [vedi lez. 5]
7. Quale comando viene utilizzato per l'inserimento e la modifica dei dati in una tabella di un database? [vedi lez. 5]
8. Che cosa significa importare/esportare dati? [vedi lez. 7]



CLIL – IN ENGLISH, PLEASE



AUDIO
Ascolta la pronuncia
del testo

ABSTRACT

VB.NET and ASP.NET

The Client/Server model on the net permits the exchange of information: the server is the computer that provides the services, the client is the computer that requests the services. Each of these two components can be static (does not reprocess the information) or dynamic when it is able to process information by activating the necessary programs. ASP.NET is a server-side programming language that lets users write dynamic pages. These are generated on the fly by the server and can therefore be different each time they are called up. Typical processing includes an HTML page, run on the client side, in which the user is required to input data and an ASP.NET page, run on the server side, which fulfils the request based on the data entered.

Processing the ASP.NET code on the server produces HTML code to send to the user's browser. ASP.NET allows the user to connect to a relational database and execute the SQL language commands needed to display and edit data in the database. The syntax of the ASP.NET language is derived from that of C#.NET.

EXERCISES

True or false?

1. ASP.NET creates static pages. T F
2. An ASP.NET program is run by the server. T F
3. The result of a query is displayed by a GridView object. T F
4. The ExecuteReader method is used to perform a query. T F
5. Cookies are used to pass information between pages. T F

Multiple choice

6. Cookies are:
 - A files stored on a server.
 - B files stored on a client.
 - C dynamic pages.
 - D static pages.
7. A session variable:
 - A expires when the user decides.
 - B does not expire.
 - C expires when the user closes the browser.
 - D expires in two hours.

GLOSSARY

ASP.NET: software development technologies for producing dynamic Internet pages.

Client: software or a computer used by the user able to access a service offered by a program running on another computer, called a server.

ExecuteNonQuery: executes an SQL statement and returns the number of rows affected.

ExecuteReader: executes an SQL statement and returns the result in a table.

Applet: small application that performs one specific task that runs inside the client.

Cookie: a small piece of data sent from a website and stored in a user's web browser while the user is browsing that website. Also known as an HTTP cookie, web cookie, or browser cookie.

Session variables: variables that exist only while the user's browser is active. They are used to store information that needs to be accessed by multiple pages in a web application.

DataReader: object used to read data from a data source, such as databases.

Read(): method to get a row from a DataReader.



GLOSSARIO
CLIL

COMPETENZE DISCIPLINARI

- Identificare e applicare le metodologie e le tecniche della gestione per progetti.
- Redigere relazioni tecniche e documentare le attività individuali e di gruppo relative a situazioni professionali.
- Interpretare i sistemi aziendali nei loro modelli, processi e flussi informativi con riferimento alle differenti tipologie di imprese.

COMPETENZE DEL XXI SECOLO**Abilità fondamentali****CULTURA SCIENTIFICA**

Capacità di usare conoscenze scientifiche e regole/modelli per interpretare un fenomeno.

Competenze trasversali**PENSIERO CRITICO**

Saper analizzare e valutare situazioni in modo da impiegare informazioni e idee per formulare risposte e soluzioni.

COMUNICAZIONE

Saper ascoltare, comprendere e contestualizzare le informazioni, per poi trasmetterle ad altri (in modalità verbale o non verbale).

COLLABORAZIONE

Saper lavorare in gruppo in vista di un obiettivo comune, prevenendo e gestendo i conflitti.

Qualità caratteriali**CURIOSITÀ**

Inclinazione a porre domande con una mentalità aperta.

OBIETTIVI FORMATIVI

- Progettare la struttura di un sito e implementarne una parte.
- Consultare fonti Internet.
- Esporre i risultati della ricerca alla classe.

TEMPI

- Definizione della struttura del sito: 1 ora.
- Implementazione di una parte significativa: 2 ore.
- Presentazione dei risultati e dibattito in classe: 1 ora.
- Autovalutazione: 10 minuti.

STRUMENTI

- Libro di testo.
- Dispositivo connesso a Internet.
- Foglio di carta.
- Ambiente di sviluppo, PowerPoint.
- Proiettore collegato al computer in classe o in laboratorio.

IL TEMA

Di seguito è riportato un estratto della prima parte della **prova di informatica** dell'Esame di Stato del 2015 per l'Istituto Tecnico, settore Tecnologico, indirizzo Informatica e Telecomunicazioni.

Abbiamo presentato il testo del tema al termine delle unità 2 e 3. Abbiamo preso in considerazione i punti 1 e 2 al termine dell'unità 2, il punto 3 al termine dell'unità 3 e i punti 4 e 5 al termine dell'unità 4. Nell'unità precedente abbiamo preso in considerazione i punti 6 e 7, per risolverli mediante la programmazione con PHP. In questa unità riprendiamo in considerazione i punti 6 e 7, per risolverli mediante la programmazione con ASP.NET.

PROGETTO E CODIFICA

[...] (Puoi vedere il testo del tema alla p. 66).

Il candidato, fatte le opportune ipotesi aggiuntive, sviluppi:

[...]

6. *il progetto della pagina dell'interfaccia web che permetta a un utente registrato di svolgere le operazioni specificate;*

Ricordiamo che: *il sito della community offre a tutti ... la consultazione dei dati online, tra cui: visualizzazione degli eventi di un certo tipo in ordine cronologico, con possibilità di filtro per territorio di una specifica provincia; visualizzazione di tutti i commenti e voti relativi a un evento;*

7. *la codifica in un linguaggio a scelta di un segmento significativo dell'applicazione web che consente l'interazione con la base di dati.*
- [...]

I nuclei tematici fondamentali e gli obiettivi

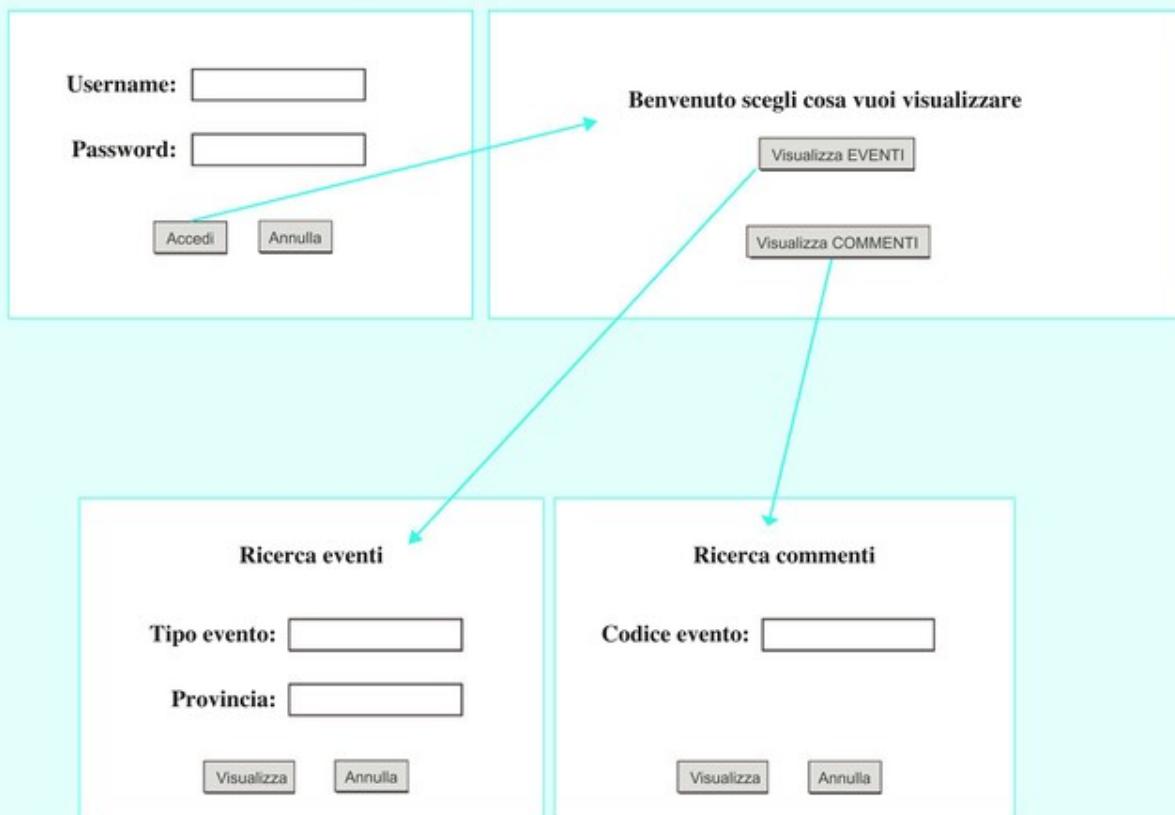
Fra i nuclei tematici fondamentali e gli obiettivi cui si riferiscono le prove d'esame (vedi la sezione finale "La preparazione alla seconda prova scritta dell'esame di Stato"), questo estratto si riferisce ai seguenti.

Nuclei tematici fondamentali	Obiettivi
<ul style="list-style-type: none"> Linguaggi per basi di dati: creazione, manipolazione e interrogazione di una base di dati. Tecnologie per il Web: i linguaggi lato client e lato server; realizzazione di applicazioni web anche con interfacciamento a basi di dati. 	<ul style="list-style-type: none"> Affrontare situazioni problematiche, utilizzando adeguate strategie cognitive e procedure operative orientate alla progettazione di soluzioni informatiche. Sviluppare applicazioni e servizi informatici per reti locali o geografiche. Scegliere sistemi e strumenti idonei al contesto proposto, in base alle loro caratteristiche funzionali. Realizzare progetti secondo procedure consolidate e criteri di sicurezza.

Svolgimento

Per quanto riguarda il punto 6 del tema d'esame possiamo progettare un sito web contenente una pagina iniziale, che permetta a un utente registrato di accedere al sito. Se l'accesso va a buon fine, si apre una seconda pagina, in cui sarà possibile scegliere il tipo di visualizzazione cui si è interessati (visualizzazione degli eventi o dei commenti relativi a un dato evento); a seconda della visualizzazione scelta, si attiva un'altra pagina, dove si possono inserire i dati di input e visualizzare i risultati.

La struttura del sito sarà la seguente:



Codifica

Per la codifica relativa al login si rimanda alla Lezione 6 di questa unità; riportiamo la codifica relativa alla visualizzazione dei commenti relativi a un dato evento.

COMMENTI.ASPX

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Commenti.aspx.cs" Inherits="Commenti" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">
    Ricerca commenti<br />
    <br />
    Codice evento
    <asp:TextBox ID="txtCodice" size='10' runat="server"></asp:TextBox>
    <br />
    <br />
<asp:Button ID="btnVisualizza" runat="server" Text="Visualizza" OnClick="btnVisualizza_Click" />
    <asp:Button ID="btnAnnulla" runat="server" Text="Annulla"
    OnClick="btnAnnulla_Click"/>
    <br />
    <asp:ListBox ID="lstCommenti" runat="server"></asp:ListBox>
    <br />
    <asp:Label ID="lblMessaggio" runat="server"></asp:Label>
    <br />
</form>
</body>
</html>
```

COMMENTI.ASPX.CS (DATABASE ACCESS)

```
using System;
using System.Data.OleDb;

public partial class Commenti : System.Web.UI.Page
{
    protected void btnVisualizza_Click(object sender, EventArgs e)
    {
        OleDbConnection connDb = new OleDbConnection();
        OleDbCommand command = new OleDbCommand();
        OleDbDataReader rsCommento;
        string codice = txtCodice.Text;
        lstCommenti.Items.Clear();
```

```

lblMessaggio.Text = "";
connDb.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;" +
    "Data Source=IDataDirectory/" + "esame2015.accdb";
connDb.Open();
string query = "Select * from commenta where cod_evento = " + codice + "";
command.Connection = connDb;
command.CommandText = query;
try
{
    rsCommento = command.ExecuteReader();
    if (rsCommento.HasRows)
    {
        while (rsCommento.Read())
        {
            string st = "voto: " + rsCommento["voto"] + " data: " + rsCommento["data"];
            st = st + " commento: " + rsCommento["commento"];
            lstCommenti.Items.Add(st);
        }
    }
    rsCommento.Close();
    connDb.Close();
}
else
    lblMessaggio.Text = "Nessun commento per l'evento " + codice;
}
catch
{
    lblMessaggio.Text = "Attenzione errore nel comando";
}
}

protected void btnAnnulla_Click(object sender, EventArgs e)
{
    txtCodice.Text = "";
    lstCommenti.Items.Clear();
}
}

```

COMMENTI.ASPX.CS (DATABASE SQL SERVER)

```

using System;
using System.Data.SqlClient;

public partial class Commenti : System.Web.UI.Page
{
    protected void btnVisualizza_Click(object sender, EventArgs e)
    {
        SqlConnection connDb = new SqlConnection();
        SqlCommand command = new SqlCommand();

```



```
SqlDataReader rsCommento;
string codice = txtCodice.Text;
lstCommenti.Items.Clear();
lblMessaggio.Text = "";

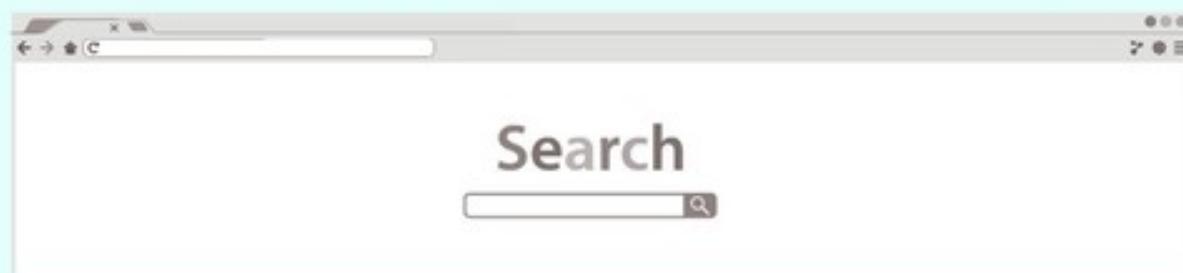
connDb.ConnectionString = "Data Source = (LocalDB)\MSSQLLocalDB;" +
    " AttachDbFileName =|DataDirectory|magazzino.mdf;" +
    " Initial Catalog = \"esame2015.mdf";
connDb.Open();
string query = "Select * from commenta where cod_evento = " + codice + "";
command.CommandText = query;
command.Connection = connDb;
try
{
    rsCommento = command.ExecuteReader();
    if (rsCommento.HasRows)
    {
        while (rsCommento.Read())
        {
            string st = "voto: " + rsCommento["voto"] + " data: " + rsCommento["data"];
            st = st + " commento: " + rsCommento["commento"];
            lstCommenti.Items.Add(st);
        }
        rsCommento.Close();
        connDb.Close();
    }
    else
        lblMessaggio.Text = "Nessun commento per l'evento " + codice;
}
catch
{
    lblMessaggio.Text = "Attenzione errore nel comando";
}
}

protected void btnAnnulla_Click(object sender, EventArgs e)
{
    txtCodice.Text = "";
    lstCommenti.Items.Clear();
}
```

COMPITI DI REALTÀ

Dopo aver confrontato la risoluzione del tema d'esame con la tua, in gruppo svolgi le seguenti attività.

- COLLABORAZIONE** Effettuate una ricerca su Internet sullo stato dell'arte dei **sistemi di gestione degli eventi** in generale e in particolare di quelli legati alle **esibizioni musicali**. Noterete probabilmente che molte realtà sono orientate alla vendita di biglietti per gli eventi. Esiste però anche un'applicazione, denominata **Local** e collegata a Facebook, che ne ha rinnovato il potenziale e soprattutto gli utenti. Analizzate la struttura e il funzionamento di questa applicazione.
- PENSIERO CRITICO** Individuate quelle che, secondo voi, sono le potenzialità dalla app Local, poi evidenziate anche i limiti (per esempio, un limite è l'interazione molto stretta con le attività svolte su Facebook, che potrebbe dare all'utente la sensazione che la app lavori senza un suo input diretto).
- COLLABORAZIONE** Raccogliete i risultati della vostra ricerca e della vostra discussione in una presentazione in PowerPoint (o in un altro strumento analogo), formata al massimo da cinque slide.
- COMUNICAZIONE** In classe, con la supervisione dell'insegnante, condividete la presentazione realizzata dal vostro gruppo con i compagni: confrontate le tematiche emerse e discutetene.
- PENSIERO CRITICO** Infine, in classe, discutete con i compagni lo svolgimento utilizzato per risolvere i quesiti del tema d'esame; poi proponete eventuali modifiche, che vi sembrino più adeguate.



AUTOVALUTAZIONE

Al termine delle attività rifletti sull'esperienza e completa la tabella di autovalutazione.

Attività	LIVELLO		
	Base	Intermedio	Avanzato
Ho compreso senza difficoltà le richieste dell'attività proposta?	Con la guida dell'insegnante e dei compagni ho compreso quasi tutte le richieste. <input type="checkbox"/>	Ho compreso le richieste e in parte le ho svolte autonomamente. <input type="checkbox"/>	Ho identificato le richieste e le ho svolte senza difficoltà. <input type="checkbox"/>
Sono riuscito a progettare il sito web?	Ho progettato il sito in modo poco dettagliato e descrivendo gli elementi in modo sommario. <input type="checkbox"/>	Ho progettato il sito tralasciando alcuni particolari. <input type="checkbox"/>	Ho realizzato uno schema completo e ben descritto del sito con le interazioni presenti tra le pagine. <input type="checkbox"/>
Sono riuscito a produrre la codifica di una parte significativa del codice?	Ho realizzato la parte HTML, ho iniziato a scrivere la codifica ASP, ma non sono riuscito a completarla. <input type="checkbox"/>	Ho realizzato la codifica di entrambe le parti, ma ci sono degli errori nell'esecuzione. <input type="checkbox"/>	Ho realizzato la codifica del sito, prelevando i dati dal database e ottenendo i risultati desiderati. <input type="checkbox"/>

8

Big Data e sistemi NoREL



PREREQUISITI

- Competenze di base di programmazione.
- Concetti fondamentali del modello relazionale.
- Conoscenza del linguaggio SQL.

PER COMINCIARE

1. Il DBMS è:

- A una parte del sistema operativo
- B un insieme di programmi per gestire una base di dati
- C un modulo del sistema operativo per memorizzare dati
- D il sistema per progettare i dati

2. Nel modello relazionale i dati sono rappresentati:

- A come insieme di vettori
- B come tabelle
- C in modo non strutturato
- D come matrici

3. Un attributo può esprimere le caratteristiche di:

- A entità
- B associazioni
- C attributi
- D entità e associazioni

4. SQL vuol dire:

- A Special Query Language
- B Structured Query Limited
- C Structured Query Language
- D Sample Query Language

CONOSCENZE

- La memorizzazione dei dati nel Cloud.
- La gestione di grosse quantità di dati non strutturati.
- I DBMS non relazionali.

ABILITÀ

- Usare un DBMS non relazionale.
- Gestire grosse quantità di dati non strutturati.

COMPETENZE

- Capacità di affrontare problematiche che richiedono l'uso di una grande mole di dati.
- Usare strumenti per gestire dati non strutturati.

1. L'APPROCCIO "CLOUD"

■ Esigenze

Negli ultimi anni si assiste a una crescita esponenziale delle apparecchiature connesse in rete e al notevole aumento della quantità di informazioni da gestire. La mole di dati da analizzare ha reso inadeguati i modelli tradizionali. Si fa strada l'esigenza di gestire risorse dati e applicazioni in remoto: all'utente finale può non interessare dove si trovino, l'importante è che funzionino. Si parla allora di un luogo generico, a cui accedere tramite la rete (che normalmente viene mostrata come nuvola, in inglese **cloud**). In pratica possiamo dire che il **cloud computing** è la possibilità di **distribuire e consumare servizi IT** attraverso il Web. Più specificatamente il cloud computing è un insieme di tecnologie informatiche che permettono di memorizzare e/o archiviare e/o elaborare dati, grazie all'utilizzo di risorse hardware e/o software distribuite sulla rete Internet attraverso un'architettura di tipo Client/Server, rendendo disponibili all'utilizzatore le risorse come se fossero implementate da sistemi (server o periferiche personali) standard. (fig. 1)

fig. 1 Cloud



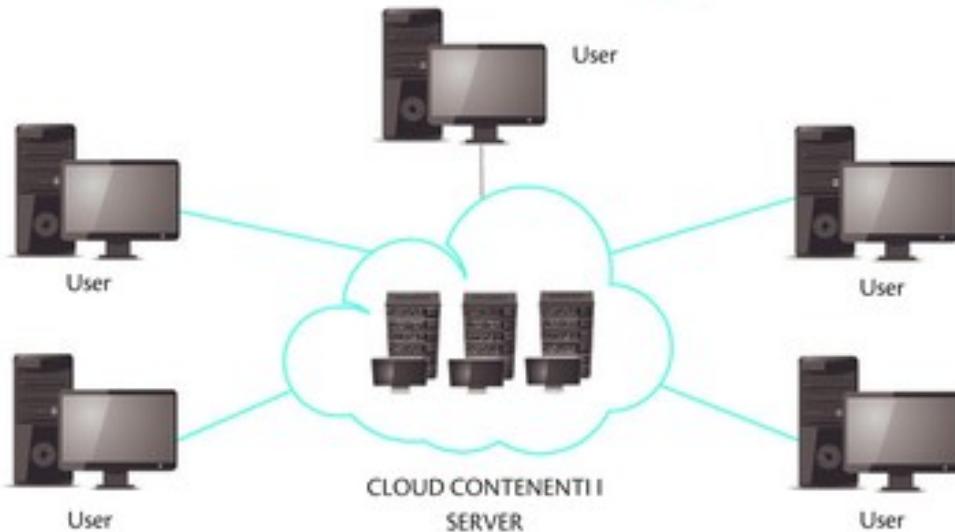
■ L'uso del Cloud

Le infrastrutture tecnologiche informatiche (hardware e software) sono sempre molto costose e richiedono tecnici specializzati per essere installate, configurate, aggiornate e protette. Una grande azienda, anche con un reparto IT esteso, non può avere a disposizione tutte le applicazioni necessarie, perché ne servono molte. Con il Cloud il cliente non deve più gestire l'hardware o il software. Di questo si occupa il fornitore del servizio, che può offrire un'infrastruttura condivisa: il cliente può usarla pagando solo l'utilizzo delle risorse impegnate. Il cliente può scegliere l'applicazione di cui ha bisogno (per esempio, quella per la gestione delle relazioni con i clienti, o per la gestione delle risorse umane o della contabilità), può personalizzarla e usarla: è sufficiente un browser e un collegamento al provider dei servizi. Queste applicazioni sono spesso integrate con App per smartphone e applicazioni collaborative in tempo reale, per esempio per comunicare con Facebook e Twitter. Spesso si fa uso delle tecnologie cloud inconsapevolmente: per esempio, la posta elettronica, ma anche la geolocalizzazione offerta dagli smartphone (grazie alla quale vengono consigliati i locali e gli esercizi commerciali più vicini) e i giochi online. Tutte le funzioni offerte dalle applicazioni sono tecnologie cloud. Con il Cloud è possibile gestire servizi concreti, come per esempio la distribuzione dei prodotti di un'azienda, le prenotazioni delle analisi mediche, il conto bancario, i servizi dell'anagrafe del Comune.

■ L'architettura

L'architettura informatica di un sistema di cloud computing prevede uno o più server, generalmente in architettura ad alta affidabilità e fisicamente collocati presso il data center del fornitore del servizio, a cui si collegano via web i client, utenti di quel cloud ([fig. 2](#)).

fig. 2 Architettura informatica di un sistema di cloud computing



Il sistema cloud richiede componenti multiple che comunicano le une con le altre su interfacce di programmazione, di solito mediante web service. In base alle varie componenti presenti, si possono realizzare diverse tipologie di servizi (o funzionalità) di cloud computing:

- **SaaS (Software as a Service)**: consiste nell'utilizzo di programmi installati su un server remoto, cioè fuori dalla rete aziendale;
- **DaaS (Data as a Service)**: con questo servizio vengono messi a disposizione via web solamente i dati. Gli utenti possono accedere tramite qualsiasi applicazione come se questi fossero residenti su un disco locale;
- **HaaS (Hardware as a Service)**: con questo servizio l'utente invia i dati in remoto, affinché vengano elaborati da computer dedicati e, successivamente, restituiti all'utente in forma di risultato dell'elaborazione;
- **PaaS (Platform as a Service)**: invece che uno o più programmi singoli, viene eseguita in remoto una piattaforma software che può essere costituita da diversi servizi, programmi, librerie e così via;
- **IaaS (Infrastructure as a Service)**: utilizzo di risorse hardware in remoto. Viene utilizzata la CPU aggiuntiva di un altro computer, o i suoi dischi, quando sorge la necessità, per esempio, di svolgere una elaborazione particolarmente complessa.

FISSA LE CONOSCENZE

1. Che cosa si intende per cloud computing?
2. Quali sono i vantaggi nell'uso di un Cloud?
3. Spiega le diverse tipologie di servizi del cloud computing.

2. BIG DATA

Negli ultimi anni si sente sempre più spesso parlare di Big Data.

Come suggerisce il nome, per Big Data si intendono grandi quantità di dati le cui sorgenti sono diverse ed eterogenee (fig. 3). I dati arrivano in maniera massiva da sorgenti come:

- sensori che raccolgono i dati del traffico;
- le informazioni meteo;
- i pacchetti GPRS dai telefoni cellulari;
- i messaggi dai social media;
- le immagini digitali e video;
- le registrazioni online delle transazioni di acquisto;
- qualsiasi altra fonte che possa produrre informazioni di interesse.

fig. 3 Provenienza dei dati nella città connessa



Di fronte a queste enormi moli di dati, i tradizionali sistemi per la gestione e l'analisi non sono più adeguati. Quindi, al fine di gestire e sfruttare questo insieme di dati eterogenei a disposizione, sono necessari nuovi modelli, nuovi paradigmi di programmazione, nuovi sistemi informatici e nuove architetture di rete. In quest'ottica, la possibilità di avere a disposizione una architettura cloud facilita notevolmente la gestione e la fruizione.

Big Data non è solo una questione di abbondanza di informazione, ma è l'occasione per trovare spunti per nuovi tipi di contenuti, per rendere le aziende più agili e capaci di rispondere a domande che prima erano considerate al di là della loro portata. I Big Data sono caratterizzati da quattro aspetti principali: Volume, Varietà, Velocità e Veridicità (Valore), conosciuti come "le quattro V dei Big Data". Esaminiamo brevemente ciò che ciascuna di esse rappresenta.

LO SAI CHE

Big Data è un termine che si riferisce all'enorme quantità di dati che viene spostata e salvata quotidianamente in rete. Possedere Big Data significa poterli analizzare e ottenere informazioni per prendere le migliori decisioni aziendali.

Volume

Gestire i Big Data comporta la capacità di acquisire, memorizzare e analizzare grandi volumi di dati. Basti pensare che il 90% dei dati di tutto il mondo è stato generato negli ultimi due anni. Si tratta di dati di tutti i tipi, alcuni dei quali hanno bisogno di essere organizzati, verificati e analizzati.

Varietà

La varietà dei tipi cui appartengono i Big Data comporta un cambiamento fondamentale nel modo in cui vengono memorizzati e analizzati. Infatti, i dati di tipo **semi-strutturato**, quali sono i Big Data, sono difficilmente gestibili con sistemi tradizionali (che funzionano molto bene con le informazioni strutturate) e richiedono pertanto tecnologie specifiche.

Velocità

La **velocità** è riferita al fatto che l'analisi dei dati deve essere fatta in tempo reale o quasi. In alcuni casi è essenziale identificare una tendenza o un'opportunità qualche minuto (o addirittura secondo) prima di un'azienda concorrente.

Veridicità (Valore)

La **veridicità** dei dati è un requisito fondamentale; solo grazie a essa i dati possono costituire **valore**, alimentando nuove intuizioni e idee. Infatti è dall'analisi che si colgono le opportunità e si trae supporto per i processi decisionali di un'azienda: più dati si hanno a disposizione, più informazioni si riescono a estrarre (ammesso che i primi siano corretti e veritieri). Il solo volume, ovviamente, non garantisce la qualità dei dati.

Per la rappresentazione e la gestione dei Big Data vengono utilizzati i cosiddetti database NoREL (o NoSQL). Si tratta, infatti, di database che devono gestire dati non strutturati, contenuti in file di caratteri difficilmente rappresentabili tramite la strutturazione rigida dei contenuti imposta dai database relazionali.

Proprio l'assenza di una rigida strutturazione nei database NoSQL ne ha determinato il successo.

Le informazioni non trovano più posto in righe elencate in tabelle, ma in oggetti completamente diversi e non necessariamente strutturati, come per esempio documenti archiviati in collezioni. Un buon esempio è MongoDB, un database NoSQL basato sui documenti: il formato del documento è una delle caratteristiche più rilevanti.

FISSA LE CONOSCENZE

1. Che cosa si intende per Big Data?
2. Spiega le "quattro V" dei Big Data.
3. Quali tipi di database vengono utilizzati per la rappresentazione dei Big Data?

3. DATABASE NON RELAZIONALI

■ DB NoREL

Uno dei problemi principali per la gestione di grosse quantità di dati non necessariamente strutturati (i Big Data) è stato dovuto alla non adeguatezza degli strumenti informatici presenti sul mercato. Circa una decina di anni fa alcuni ricercatori proposero l'uso di strumenti non relazionali per gestire le grosse moli di dati che la rete metteva a disposizione. In particolare nel 1998 l'italiano Carlo Strozzi coniò il termine NoSQL, acronimo di *Not only SQL*, usato per identificare tutti quei database che non fanno uso di un modello di dati relazionale. Lo sviluppo prende slancio però dal 2009, quando molte aziende decidono di passare a questo tipo di approccio.

Due sono state le principali esigenze che hanno spinto le aziende a ricercare una soluzione alternativa ai database relazionali:

- la continua crescita del volume di dati da memorizzare;
- la necessità di elaborare grandi quantità di dati in poco tempo.

Tra i pionieri troviamo Google, con il suo BigTable, e Amazon, con DynamoDB. Entrambi i DB hanno dimostrato la necessità di uscire dai classici schemi relazionali per permettere la scalabilità dei loro servizi.

Un database non relazionale si differenzia dal modello tradizionale, non solo perché non segue la teoria relazionale, ma perché ha un approccio completamente diverso da quello tipico della teoria dei database classica. Infatti la caratteristica principale è la mancanza sia di tabelle sia dello schema logico, abbinata al fatto che per collegare i dati non è necessario utilizzare le operazioni join: si parla di schema implicito nei dati stessi.

Normalmente si usano i termini **NoREL** e **NoSQL** in modo abbastanza interscambiabile.

Vediamo meglio le differenze tra il modello relazionale e quello non relazionale, aiutati dallo schema mostrato nella tabella.

	Database relazionali	Database non relazionali
ORGANIZZAZIONE MEMORIZZAZIONE DATI	Basato su tabelle. Ogni riga corrisponde a un record.	Esistono specifiche soluzioni di memorizzazione in base alle differenti esigenze (es.: dati gerarchici, documenti, ecc.).
SCHEMA LOGICO	Schema predefinito. Si conosce l'organizzazione dei dati e si sa cosa si può fare e che risultati attendersi. Esistono dei vincoli sui dati stessi. La modifica dello schema per aggiungere o modificare attributi può essere pesante. È necessario, in questo caso, bloccare tutte le attività, e provvedere alla riorganizzazione di tutto il database.	Non esiste lo schema, per cui non si sa a priori come sono organizzati e che cosa si può ottenere. La modifica è semplice, perché lo schema è隐式 nei dati, per cui è possibile modificare e aggiungere documenti, potendo continuare a lavorare sugli altri dati.
SCALABILITÀ	Scalabilità verticale. All'aumento della dimensione del database si incrementa la potenza dell'hardware.	Scalabilità orizzontale. All'aumento delle dimensioni dei dati si aggiungono nuovi server in cui andranno collocati i nuovi dati.

	Database relazionali	Database non relazionali
LINGUAGGIO	L'accesso ai dati avviene tramite il linguaggio SQL. È un linguaggio strutturato molto potente per definire e manipolare i dati.	Poiché non c'è lo schema, esistono differenti modalità di memorizzazione in base ai differenti tipi di dati, esistono anche differenti linguaggi specifici.
INTERROGAZIONI	Query complesse basate sui join.	Interfacce non standard per le query, che sono semplici, e assenza di join.

Esistono attualmente anche delle soluzioni ibride: alcuni DBMS relazionali che implementano delle funzionalità del mondo NoREL (per esempio, la scalabilità orizzontale) e DBMS NoREL che introducono funzionalità del mondo relazionale (per esempio, il controllo dello schema).

■ Tipi di NoSQL database

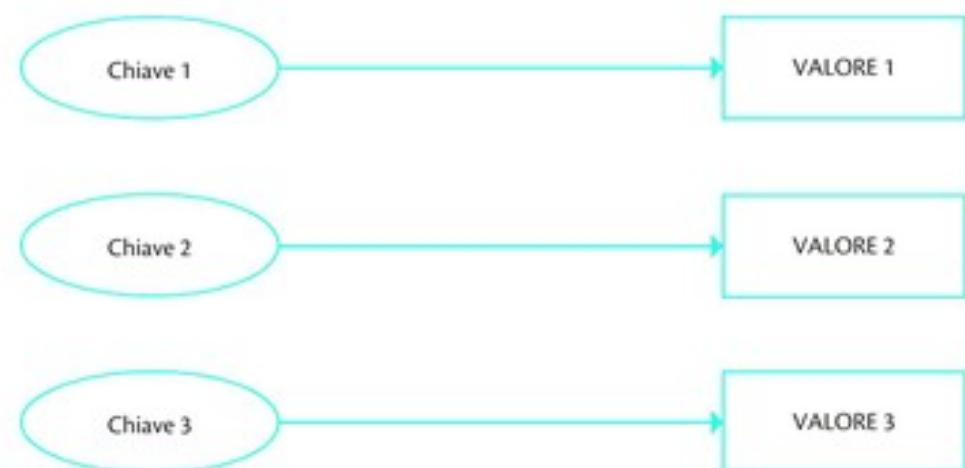
Esistono quattro differenti famiglie di DBMS NoSQL.

1. **Key-Value:** definiti da un semplice dizionario (o mappa) che permette all'utente di recuperare e aggiornare un valore memorizzato conoscendone la chiave.
2. **Column-Oriented:** sistemi che, pur utilizzando le tabelle, memorizzano informazioni non per riga, ma per colonna.
3. **Graph:** rappresentano la realtà sotto forma di nodi e rami di un grafo. Ai nodi, come ai singoli rami, vengono associate le informazioni attraverso la logica Key-Value.
4. **Document-Oriented:** memorizza le informazioni come collezioni di documenti. Un documento ha un formato riconosciuto (JSON, XML, etc.), che permette poi al server di eseguire delle query sui dati.

Key-Value

È la più semplice modalità di gestione dei dati di tipo NoSQL. In pratica è un dizionario piatto con delle chiavi che vengono associate a dei valori. Nel sistema vi sono quindi coppie di elementi: la chiave e il valore a essa associato (fig. 4). Quest'ultimo contiene l'informazione vera e propria, che può essere qualunque cosa, mentre la chiave è l'identificatore che permette l'estrazione rapida del valore dal database.

fig. 4 Sistema Key-Value



Questa modalità nasce dall'esigenza di Facebook di permettere un facile accesso ai contenuti. Non esiste quindi una struttura: man mano che i contenuti aumentano, si aggiungono nuovi server (scalabilità orizzontale). È un modello molto semplice e molto veloce, che offre la possibilità di effettuare ricerche rapide di singoli blocchi di informazione. I migliori contesti in cui sono utilizzati questi tipi di database NoSQL sono quelli in cui si trattano valori o collezioni a cui è facilmente associabile un identificatore univoco da usare come chiave.

Tra i principali sistemi di questo tipo troviamo Redis, Riak, Memcached.

Database Column-Oriented

Un Column-Oriented DBMS è un sistema che memorizza i dati non come righe di tabelle, ma come colonne ([fig. 5](#)).

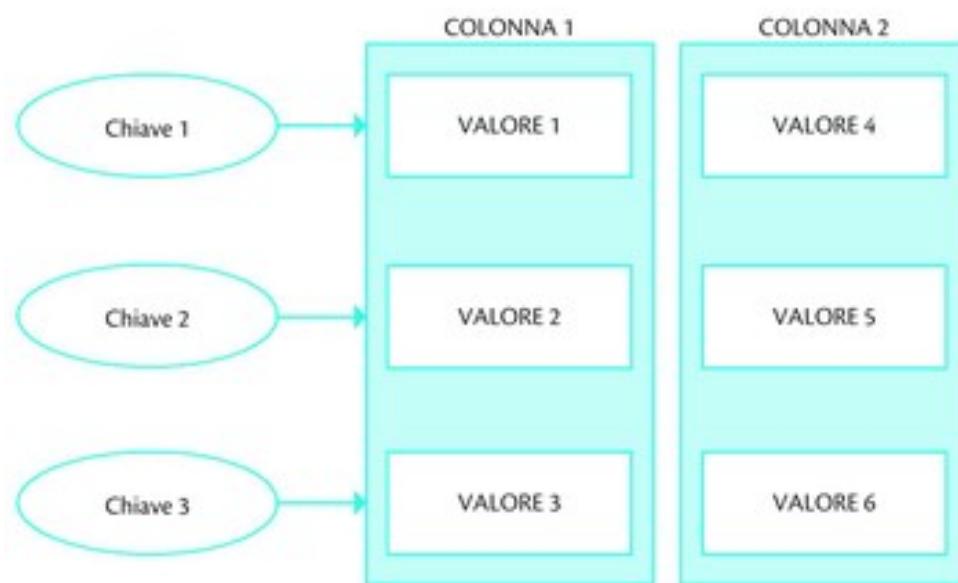


fig. 5 Sistema Column-Oriented

Una colonna può essere formata da attributi complessi. Per quanto riguarda la struttura dei dati, in questi sistemi ogni valore può essere una stringa o, se l'implementazione lo permette, qualsiasi altro tipo di dato primitivo. Non avendo uno schema formale, non c'è nessuna specifica che indichi attributi obbligatori e il loro tipo.

In questa tipologia di BD i dati, come indica il nome stesso, sono memorizzati insieme per colonna, mantenendo così la classica organizzazione in righe e colonne. Per evitare la presenza di dati *null*, i database orientati a colonna permettono a ogni riga di avere un set diverso di colonne, che possono essere aggiunte se necessario o tolte se inutilizzate.

I vantaggi offerti da questo tipo di orientamento si vedono soprattutto in lettura: le query mirano a recuperare valori soltanto da determinate colonne e quindi non da tutta la riga. Essendo composte da tipi di dati uniformi, le colonne risultano essere più facili da comprimere, a vantaggio della velocità di esecuzione.

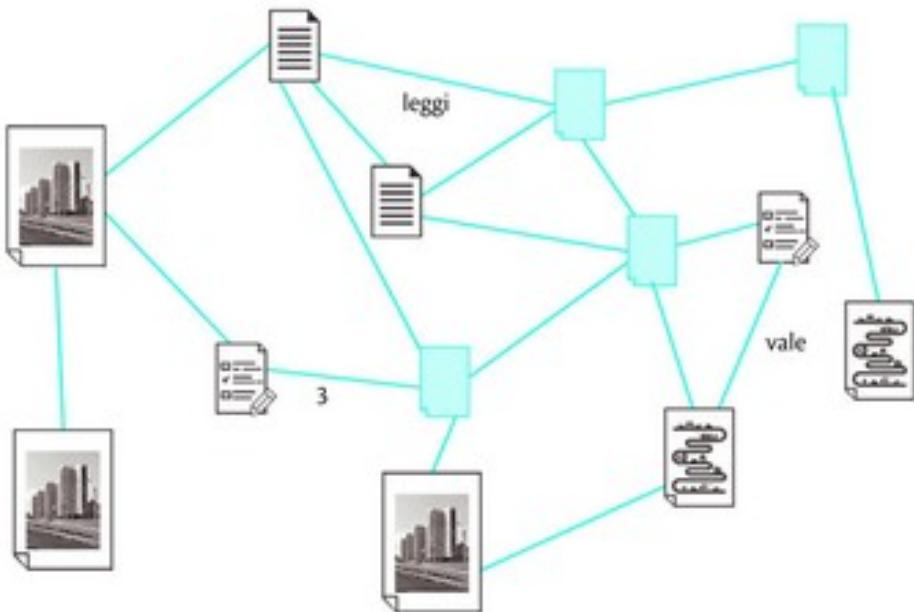
L'accesso e la memorizzazione possono avvenire in modo simile a come avviene nella logica Key-Value.

Tra i principali sistemi di questo tipo troviamo Cassandra, HBase, HyperTable, Amazon DynamoDB.

Graph

Si basa sulla teoria dei grafi, strutture dati che rappresentano una rete di nodi collegati tra loro da archi. Le informazioni possono essere custodite sia nei nodi sia negli archi (fig. 6). La forza di questa tipologia è nell'insieme del valore informativo che si può estrapolare ricostruendo percorsi attraverso il grafo.

fig. 6 Sistema Graph



È molto usato per memorizzare informazioni dove il legame tra oggetti è molto forte. Ne esistono molte applicazioni nel mondo reale. Nasce per rispondere alle necessità dei Social Network, dove troviamo molte relazioni tra le persone: se due utenti sono "amici", possiamo rappresentarli come oggetti messi in relazione tra loro. In questo modo diventa semplice navigare il grafo e recuperare gli "amici degli amici", proponendo così nuove potenziali conoscenze a un utente.

Un altro esempio può essere la gestione di una rete di trasporti, dove ogni nodo rappresenta una città e il collegamento tra due nodi contiene la distanza tra di esse. Si può così calcolare, per esempio, la distanza tra due città non direttamente collegate, sommando la distanza di ogni tratta. In generale, è possibile farlo, definendo un punto di partenza e uno di arrivo. È possibile, per esempio, calcolare la distanza totale e ogni altra grandezza derivata (come tempi, costi, eccetera).

Tra i principali sistemi di questo tipo troviamo Neo4J, Infinite Graph, OrientDB.

Document-Oriented

Nasce per memorizzare e ricercare dei documenti. È forse la tipologia più diffusa e usata.

Può essere vista come estensione del Key-Value, dove il valore è una struttura simile a un oggetto, detto documento. Ciascun documento, che può essere qualcosa di molto complesso, possiede un certo numero di proprietà, che rappresentano le informazioni vere e proprie. Può essere una lista di elementi, una mappa, una gerarchia di sottodocumenti,

qualunque oggetto (fig. 7).

La natura stessa dei documenti è eterogenea. I documenti non hanno una struttura fissa, anche se tutti avranno delle proprietà spesso simili. Per esempio, se i documenti devono rappresentare degli studenti, probabilmente vi saranno in tutti delle proprietà classiche (nome, cognome, data di nascita); ma si può decidere di inserire in un documento un numero di telefono, mentre in un altro verrà inserito un indirizzo e-mail, in base alle informazioni presenti.

Volendo fare un parallelo col mondo relazionale, possiamo vedere il documento come l'equivalente del record delle tabelle.

Poiché i documenti possono essere facilmente messi in relazione tra loro con dei riferimenti, è possibile rappresentare agevolmente delle gerarchie.

Tra i principali sistemi di questo tipo troviamo MongoDB, CouchDB, RavenDB

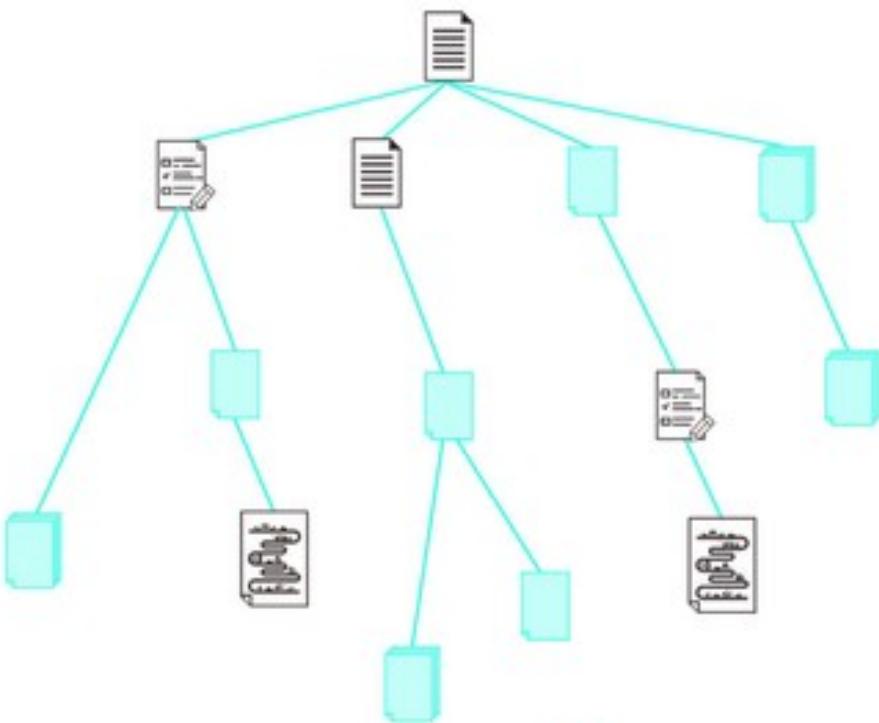


fig. 7 Sistema Document-Oriented

Hadoop

Una particolare attenzione merita Hadoop: tecnologia associata al movimento dei Big Data degli ultimi anni.

Apache Hadoop è un framework che supporta applicazioni distribuite. Hadoop permette alle applicazioni di lavorare con migliaia di nodi e grandi quantità di dati, mettendo a disposizione librerie di semplice utilizzo. È una tecnologia sviluppata da Google (ispirata dalla MapReduce di Google e dal Google File System) e dalla comunità open source, che ha rivoluzionato le percezioni delle aziende su cosa può essere fatto con i loro dati, dalla dimensione alla varietà. La crescita esplosiva associata al crollo dei costi dei server e quella delle infrastrutture cloud ha fatto sì che le tecnologie Hadoop siano diventate il punto di riferimento per le medie e grandi aziende.

FISSA LE CONOSCENZE

1. Che cosa si intende per database NoREL?
2. Quali sono i tipi di DBMS NoSQL?
3. Che cosa si intende per gestione Key-Value?
4. Descrivere le principali differenze tra DBMS NoREL e DBMS relazionali.

4. GLI OPEN DATA

■ Cosa sono gli Open Data

Gli Open Data, definiti anche con il termine italiano di **dati aperti**, sono informazioni digitali generalmente messe a disposizione da enti governativi per finalità di trasparenza e partecipazione diretta alla pubblica amministrazione (e-government), liberamente accessibili a tutti i cittadini tramite la rete Internet.

LO SAI CHE

La maggior parte delle fonti rilasciano i contenuti sotto licenza Creative Commons (CC BY 3.0 IT) (<https://tinyurl.com/ybynuuk6>) o Italian Open Data License v2.0 (<https://tinyurl.com/y9njf88v>).

La definizione di **dati aperti** rientra in un più ampio movimento di pensiero globale che mira a difendere i diritti dei cittadini e presuppone quindi che tutti possano essere in grado di usare, riutilizzare e ridistribuire i dati: non ci devono essere discriminazioni né in ambito di iniziativa né contro soggetti o gruppi. Per esempio, la clausola *non commerciale*, che vieta l'uso a fini commerciali o restringe l'utilizzo solo per determinati scopi (per esempio, quello educativo) non è ammessa.

Gli Open Data possono riferirsi alle più svariate tematiche, come dati anagrafici, governativi, sociali, economici, scientifici, medici. È consentita la consultazione, divulgazione, manipolazione da parte di tutti, con l'obbligo di citare la fonte di provenienza, di mantenere i dati sempre aperti e di non alterare i contenuti originali.

Esistono però alcune difficoltà oggettive che impediscono una ampia diffusione dei dati aperti su larga scala: alcune sono legate al concetto di tutela della privacy (in Italia regolamentato dal DL n. 196/2003, adeguato alle disposizioni del Regolamento UE 2016/679 tramite il DL n. 101/2018), altre al concetto di valore dei dati. Per molte organizzazioni, pubbliche e private, i dati sono considerati come un vero patrimonio economico di ingente valore, quindi da tutelare e non condividere.

■ Caratteristiche dei dati aperti

In estrema sintesi si possono individuare alcuni aspetti fondamentali che caratterizzano un insieme di dati come aperto:

- i dati devono essere indicizzati dai motori di ricerca e quindi reperibili nelle ricerche web;
- i dati devono essere disponibili in un formato aperto, standardizzato e leggibile da un'applicazione informatica per facilitare la loro consultazione e incentivare il loro riutilizzo anche in modo creativo;
- i dati devono essere rilasciati attraverso licenze libere che non impediscono la diffusione e il riutilizzo da parte di tutti i soggetti interessati.

Gli Open Data sono generalmente strutturati sotto forma di **database** (.mdb, .sql...) e/o **fogli elettronici** (.xlsx, .odt, .csv...) e possono quindi essere gestiti con i più comuni software in commercio. Inoltre è possibile interfacciarli con pagine web dinamiche (front-end) e quindi servire da motore per la gestione dei dati (back-end).

Nel panorama nazionale alcuni enti pubblici forniscono grandi quantità

di Open Data. Tra questi quelli che risultano maggiormente ricchi di informazioni sono i portali:

- INPS (<https://tinyurl.com/y9f55y6z>) (fig. 8);
- ISTAT (<https://tinyurl.com/y72uapf2>);
- MIUR (<https://tinyurl.com/ycufw9ts>) (fig. 9);
- Pubblica amministrazione (<https://tinyurl.com/ycswfai5>).

fig. 8 La pagina Open Data del portale INPS

The screenshot shows the INPS Open Data section. At the top, there's a navigation bar with links to 'Istituto', 'Dati, ricerche e bilanci', 'Avvisi, bandi e fatturazione', 'INPS Comunica', 'Prestazioni e servizi', 'Amministrazione trasparente', 'Lingua ITA', 'Assistenza', and 'Contatti'. Below the navigation is a search bar with placeholder text 'Cerca un servizio, prestazione, informazione! Es. Pensioni' and a 'Cerca' button. The main content area has a title 'Scarica gli Open Data INPS' with a document icon. It features two sections: 'Open Data' on the left with a search bar and a sidebar for 'Bilanci e rendiconti', 'Bilancio Sociale', 'Casellario dell'assistenza', and 'Note trimestrali sulle tendenze dell'occupazione'; and 'Open Data più utilizzati' on the right, which lists three datasets: 'Importo contributi lavoratori domestici italiani e stranieri. Coefficient...', 'Legge 104 e congedo straordinario. Numero beneficiari per tipologia cont...', and 'Importo contributi lavoratori domestici italiani e stranieri. Coefficient...'. Each dataset card includes a 'Dataset' icon, a brief description, and a 'Dettagli' button. At the bottom, there are filters for 'Tutti gli Open Data', 'Argomento', 'Fonte', and 'Periodo', along with 'Visualizzazione' buttons for 'Griglia' (selected) and 'Lista'.

The screenshot shows the MIUR Open Data page for the 'Portale Unico dei Dati della Scuola'. The top navigation bar includes 'IL PROGETTO', 'OPEN DATA' (selected), 'ESPLORA I DATI', 'DOCUMENTALE', and 'APPROFONDIMENTI'. A search bar at the top right contains 'Cerca nel dataset' and a 'CERCA' button. The main content area has a title 'Open Data' with a background image of a classroom. On the left, a sidebar for 'Catalogo Dataset' includes links for 'Ambito Scuola', 'Ambito PON', 'Interroga i dati con SPARQL', 'Ambito Scuola', and 'Ambito PON'. The main text area discusses the value of open data for schools, mentioning the valorization of the information patrimony and the promotion of transparency and participation in school improvement. It also highlights the creation of new services for students, teachers, families, research institutions, and anyone interested in the world of schools.

fig. 9 Portale del MIUR per gli Open Data

Ultimamente anche molti Comuni e Regioni stanno procedendo alla pubblicazione dei dati in chiaro sui propri portali istituzionali per sostenere i processi di trasparenza, democrazia e stimolo alla nuova imprenditoria.

ESEMPIO

Consideriamo adesso un esempio di utilizzo degli Open Data seguendo la seguente procedura: 1. ricerca della fonte; 2. scelta del formato; 3. download; 4. importazione tramite software; 5 analisi dei dati.

A tal fine utilizziamo, come fonte, il portale istituzionale dell'Istituto Nazionale di Previdenza Sociale (INPS). L'URL da utilizzare è quello menzionato precedentemente.

Il sito mette a disposizione molte voci suddivise per categorie oppure l'utilissima casella di ricerca tramite stringa di testo e una ricerca avanzata con filtri. Dopo aver scelto la voce di interesse bisogna stabilire il formato da scaricare; questo dipende dal passaggio successivo, ovvero da quale software utilizzeremo per l'importazione.

The screenshot shows the INPS website's Open Data section. At the top, there's a navigation bar with links like 'Dati, ricerche e bilanci', 'Avvisi, bandi e fatturazione', 'INPS Comunica', 'Prestazioni e servizi', 'Amministrazione trasparente', and language and contact options. Below the header, a search bar and a 'Cerca' button are visible. The main content area has a large heading 'Scarica gli Open Data INPS'. On the left, a sidebar lists various datasets: 'Bilanci e rendiconto', 'Bilancio Sociale', 'Casellario dell'assistenza', 'Note trimestrali sulle tendenze dell'occupazione', 'Open Data' (selected), 'Applicazioni realizzate', 'Diffusione Opendata', 'Il percorso Inps per gli Open Data', 'Scarica gli Open Data INPS' (link), 'API INPS', 'LOD INPS', 'JSON INPS', and 'Operazione porte aperte'. The 'Open Data' section is expanded, showing its details: 'Composizione del reddito lordo disponibile delle famiglie consum.', 'Data di creazione: 14/02/2014', 'Proprietario del dato: INPS', 'Licenza: ©Italian Open Data License 2.0', and a table of 'File allegati' with columns 'Formato', 'Download', and file names (XLS, CSV, XML, OWL, JSON) with their respective file sizes (108, 60, 66, 61, 61).

fig. 10 Informazioni sul dataset messo a disposizione dal portale INPS

Se la scelta ricade sul formato .xls basterà semplicemente scaricare il file ed eseguirlo con un programma di foglio elettronico (es. Microsoft Excel, oppure Open Office Calc) per visualizzarne il contenuto. Se invece si utilizza un formato non proprietario (per esempio .csv), in genere file di testo delimitato da caratteri o spazi, bisognerà prima avviare una procedura di importazione, con il vantaggio che sarà possibile usare una più ampia categoria di software, come per esempio i database.

Nel nostro esempio utilizziamo Microsoft Excel per importare un file .csv.

La procedura è la seguente.

1. Dalla finestra di dialogo Apri del programma impostare il formato file .csv, come è indicato nella figura (in basso a destra).
2. Selezionare il file dalla cartella in cui è stato effettuato il download ([fig. 11](#)).



[fig. 11](#) Informazioni sul file CSV scaricato

3. Se vogliamo vedere il contenuto "grezzo", possiamo aprire il file con il blocco note ([fig. 12](#)).

A screenshot of a Windows Notepad window titled 'ID-1779 - Blocco note'. The window contains the following text:

```
Redditi;2008;2009;2010;2011;2012
Reddito primario lordo ;1.188;1.136;1.145;1.165;1.148
Imposte correnti sul reddito e patrimonio;-188;-182;-187;-188;-198
Contributi sociali netti;-249;-245;-248;-251;-252
Prestazioni sociali Inps;263;276;285;291;299
Altre prestazioni sociali;45;46;46;48;47
Altri trasferimenti netti ;-10;-9;-10;-13;-13
Reddito disponibile lordo ;1.049;1.021;1.031;1.052;1.030
```

[fig. 12](#) Il contenuto "grezzo" sul file CSV scaricato

4. Confermare l'apertura del file: si otterranno i dati aperti nel foglio elettronico (in questo caso senza alcuna formattazione) ([fig. 13](#)).

	A	B	C	D	E	F	G
1	Redditi	2008	2009	2010	2011	2012	
2	Reddito pr.	1.188	1.136	1.145	1.165	1.148	
3	Imposte co.	-188	-182	-187	-188	-198	
4	Contributi	-249	-245	-248	-251	-252	
5	Prestazion	263	276	285	291	299	
6	Altre prest	45	46	46	48	47	
7	Altri trasfe	-10	-9	-10	-13	-13	
8	Reddito di	1.049	1.021	1.031	1.052	1.030	
9							
10							

fig. 13 I dati importati al termine della procedura

La stessa procedura può essere replicata con innumerevoli file e le ultime versioni dei software forniscono procedure sempre più complete per l'importazione di svariati formati alternativi (per esempio XML).

■ Applicazione degli Open Data

Gli Open Data stanno divenendo risorse informative sempre più utili in ambito lavorativo, didattico e di sviluppo.

Ambito lavorativo

Gli Open Data rappresentano, in molte occasioni, il nucleo basilare per usufruire di preziose informazioni, che permettono l'aggiornamento di software e portali web (si pensi alle banche dati per un software di gestione delle ultime norme in ambito legislativo, oppure ai dati statistici sui consumi per le attività commerciali).

Ambito didattico

Gli Open Data rappresentano ottime risorse per tracciare percorsi orientati all'acquisizione di competenze per il curriculum informatico (foglio elettronico e progettazione di database), dando la possibilità agli studenti di cimentarsi in situazioni e problemi vicini alla realtà, rappresentati in una mole di dati ricca e stimolante.

Ambito di sviluppo

I governi nazionali stanno sempre più investendo tempo e denaro per evidenziare i benefici a livello sociale che derivano dal rilascio di questi dati. Rendere disponibili le informazioni come dati aperti può infatti

aumentare la comprensione dei costi in campo sanitario e dei risultati raggiunti in termini di welfare e tutela dei consumatori. Inoltre stanno aumentando anche gli incentivi economici. Tramite bandi e concorsi viene stimolato l'avvio di start-up innovative che operano nell'ambito tecnologico, manipolando e gestendo gli Open Data, e sviluppano applicazioni per renderne fruibili nel migliore dei modi i contenuti.

Nella **fig. 14** viene mostrato il portale degli Open Data della pubblica amministrazione.

fig. 14 Il portale degli Open Data della pubblica amministrazione

The screenshot shows the homepage of the dati.gov.it portal. At the top, there's a navigation bar with links for Dati, Notizie, Fare Open Data, Federazione dei cataloghi, Monitoraggio, Sviluppatori, and Contatti. Below the navigation is a search bar with placeholder text "Cerca tra i dati della pubblica amministrazione" and a magnifying glass icon. The main content area is titled "Naviga i dati per categoria tematica:" and features seven rows of icons representing different thematic categories: Agricoltura, pesca, silvicultura e prodotti alimentari; Economia e finanze; Istruzione, cultura e sport; Energia; Ambiente; Governo e settore pubblico; Salute. Below these are six more categories: Tematiche internazionali; Giustizia, sistema giuridico e sicurezza pubblica; Regioni e città; Popolazione e società; Scienza e tecnologia; and Trasporti. A blue callout box at the bottom left is titled "FISSA LE CONOSCENZE" and contains three numbered questions:

1. Da chi sono accessibili gli Open Data?
2. Quali sono i principali formati disponibili per gli Open Data dell'INPS?
3. Quali sono gli ambiti in cui si stanno applicando maggiormente i Big Data?



RIPASSIAMO INSIEME

Il Cloud

Cloud computing è un insieme di tecnologie informatiche che permettono di memorizzare e/o archiviare e/o elaborare dati. Prevede l'utilizzo di risorse hardware e/o software distribuite sulla rete Internet attraverso un'architettura di tipo Client/Server e rende disponibili all'utilizzatore le risorse come se fossero implementate da sistemi standard. L'architettura informatica di un sistema di cloud computing prevede uno o più server reali, a cui si collegano via web gli utenti di quel sistema.

Big Data

Per Big Data si intendono grandi quantità di dati **non strutturati** e provenienti da numerose **sorgenti eterogenee**. Possedere Big Data significa poterli analizzare per ottenere le informazioni necessarie per prendere le migliori decisioni aziendali. Un Big Data è caratterizzato da quattro aspetti

principali (le quattro "V"): Volume, Varietà, Velocità e Veridicità (Valore).

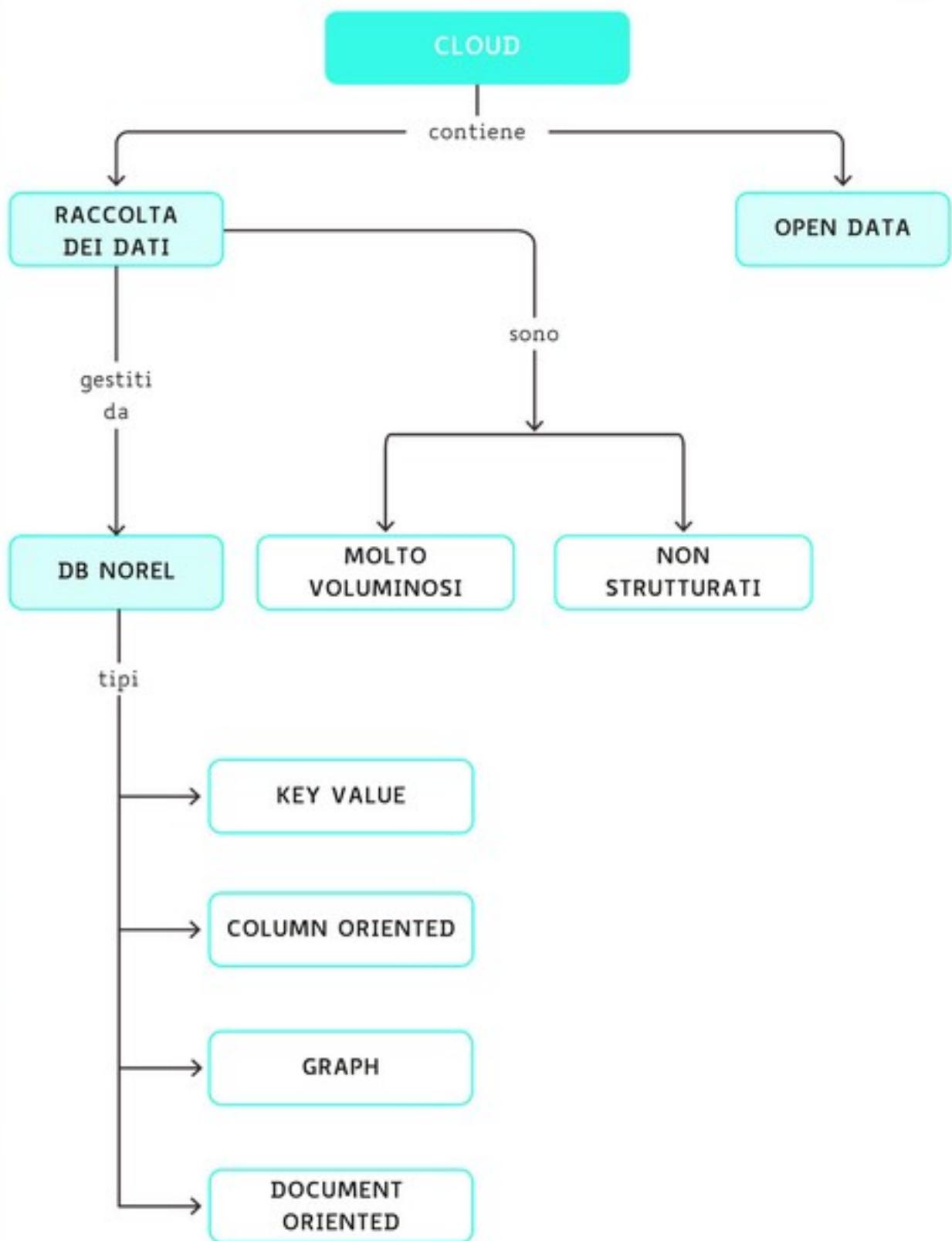
Database non relazionali

I database non relazionali (NoREL o NoSQL) hanno un approccio completamente diverso da quello previsto dalla teoria dei database classica: non lavorano su tabelle strutturate, ma su dati di diversa natura e non strutturati. Esistono diverse famiglie di DBMS NoSQL: Key-Value, z, Graph, Document-Oriented. Alcuni sistemi NoREL presenti sul mercato sono: Redis, Riak, Infinite Graph, OrientDB, MongoDB, CouchDB, RavenDB, Cassandra, HBase, Hypertable, Amazon DynamoDB.

Gli Open Data

Gli Open Data sono le informazioni digitali messe a disposizione dagli **enti governativi** per la **trasparenza** e per la **partecipazione** dei cittadini alla pubblica amministrazione.





TEST**TEST**

Svolgi il test interattivo

Vero o falso?

1. Nel cloud computing posso avere in remoto solo i dati e non le applicazioni. F
2. I database NoSQL non definiscono gli schemi logici. F
3. Gli Open Data sono informazioni strettamente riservate. F
4. MongoDB è un DBMS molto utilizzato per la gestione dei documenti. F
5. Una delle principali motivazioni che ha portato alla nascita dei sistemi NoSQL è la continua crescita del volume di dati da memorizzare. F

Scelta multipla (una sola è la risposta esatta)

6. Quale tra queste non è una delle 4 V dei Big Data?
 A Volume.
 B Visibilità.
 C Velocità.
 D Veridicità.
7. Quale di queste non è una delle famiglie di DBMS NoSQL?
 A Graph.
 B Key-Value.
 C Row-oriented.
 D Document-oriented.
8. Quali tra queste è una delle principali caratteristiche dei DBMS NoSQL?
 A Avere la scalabilità verticale.
 B Essere basato su tabelle.
 C Avere SQL come unico linguaggio di interrogazione.
 D Non avere uno schema logico, se non隐式的.
9. NoSQL è l'acronimo di:
 A not only SQL.
 B not use SQL.
 C not SQL.
 D natural organization is SQL.
10. Quale non è una caratteristica degli open data (dati aperti)?
 A Essere indicizzati.
 B Essere utilizzabili dalle applicazioni informatiche più comuni.
 C Essere disponibili in formati proprietari.
 D Essere rilasciati tramite licenze libere.
11. L'utilizzo di programmi installati su un server remoto, cioè fuori dalla LAN aziendale, si chiama:
 A SaaS
 B DaaS
 C HaaS
 D AaS

PREPARATI PER IL COLLOQUIO ORALE

1. Su quale architettura si basa il concetto di cloud computing? [vedi lez. 1]
2. Quali sono le quattro V dei Big Data? [vedi lez. 2]
3. Descrivi come sono organizzati i database Key-Value [vedi lez. 3]
4. Che cosa si intende per dato aperto? [vedi lez. 4]



CLIL – IN ENGLISH, PLEASE



AUDIO
Ascolta la pronuncia
del testo

ABSTRACT

Cloud computing

Cloud computing is a set of computing technologies that permit the use of virtual hardware and software resources deployed remotely in a typical Client/Server architecture.

It make possible to deploy and consume data and information over the Web.

Big Data means to manage a big amount of not structured data coming from many heterogeneous sources. The use of Big Data means to be able to decide in the best way for the company. A Big Data has 4 principal features: Volume, Speed, Variety and Value. NoREL databases has a different way to manage data. They don't use structured tables and manage data coming from different media. They are also called DB NoSQL.

We can find different types of NoSQL DBMS: Key-Value, Column-Oriented, graph, Document-Oriented.

Open data are digital informations that are deployed from government bodies to be used from everybody in a free way.

EXERCISES

True or false?

1. In the cloud computing architecture the data are stored locally T F
2. MongoDB is a NoREL DBMS based on Document Management T F
3. The type of the Big Data is eterogeneous T F
4. Open Data must be deployed in an open format T F
5. The typical architecture of cloud computing is peer-to-peer T F

Multiple choice

6. NoSQL means:
 - A not only SQL.
 - B not use SQL.
 - C no SQL.
 - D natural organization is SQL.
7. NoSQL DBMS are based on an architecture:
 - A network.
 - B key-value.
 - C row-oriented.
 - D table.
8. Which is not a feature of Big Data?
 - A Volume.
 - B Variety.
 - C Value.
 - D Vaquity.

GLOSSARY

Big Data: a big amount of not structured data.

Cloud computing: computing technologies that use virtual hardware and software resources.

MongoDB: one of the NoSQL database most used.

NoREL: synonymous of NoSQL.

NoSQL: system that allows the management of Big Data.

Open Data: free data deployed from government bodies.



GLOSSARIO
CLIL

ALTERNANZA SCUOLA - LAVORO

Percorsi per le competenze trasversali e per l'orientamento

Lezione 1

I percorsi per le
competenze trasversali
e per l'orientamento

Lezione 2

Riconoscere e valutare
le competenze trasversali

Lezione 3

Esempio di una relazione
conclusiva sui *Percorsi*

1. I PERCORSI PER LE COMPETENZE TRASVERSALI E PER L'ORIENTAMENTO

■ LE FINALITÀ DEI PERCORSI

Nel 2015 una legge apposita (legge 13 luglio 2015 n. 107) ha reso parte integrante del percorso scolastico nel secondo ciclo del sistema di istruzione e formazione i percorsi di alternanza scuola-lavoro. Successivamente tali percorsi sono stati chiamati "Percorsi per le competenze trasversali e per l'orientamento". In ogni caso si cerca, soprattutto nel mondo della formazione tecnica, di far acquisire ai giovani come te competenze che possano essere spese nel mercato del lavoro. Hai cominciato a fare delle attività in collaborazione e cooperazione con le aziende fin dal terzo anno di questo corso di studi, e per te è stata un'opportunità per cominciare a conoscere più da vicino il mondo del lavoro prima di concludere il percorso scolastico. Ti ha permesso tra l'altro di conoscere le realtà produttive presenti nel tuo territorio, di trovare nuove motivazioni allo studio e di capire come funziona il mondo del lavoro. In quest'ultimo anno il percorso si completa e ti permetterà di fare ulteriori esperienze, che ti aiuteranno, tra l'altro, a orientarti rispetto alle scelte professionali che dovrai fare al termine del percorso scolastico. Riconsideriamo quanto hai fatto in questi anni e le modalità con cui hai potuto svolgere le attività. Prova a confrontare quanto diremo con ciò che hai realmente svolto, in modo che tu sia in grado di consolidare l'esperienza fatta e raccontarla a un futuro colloquio di lavoro.

■ L'ORGANIZZAZIONE DEI PERCORSI

Le diverse modalità di organizzazione

In questi anni hai potuto svolgere le attività secondo vari modelli organizzativi. Il tuo istituto scolastico, in collaborazione con aziende, ha predisposto per te un percorso formativo personalizzato, coerente con il tuo indirizzo di studi. All'inizio dell'esperienza di alternanza sono stati stabiliti i criteri che sono serviti a valutare le tue attività e l'efficacia del processo formativo. Hai alternato periodi di attività a scuola con periodi in cui sei stato a stretto contatto con persone provenienti dal mondo del lavoro, sia presso la loro azienda (visitandola o lavorandoci) sia imparando da loro a scuola. Al termine di questo percorso di formazione le competenze da te acquisite sono state definite dal Consiglio di classe redigendo la **certificazione delle competenze**.

Tra le varie figure con cui hai avuto a che fare nel corso di tale esperienza, due in particolare sono state fondamentali: il **tutor scolastico** e quello **aziendale**. Il primo, in collaborazione con il Consiglio di classe, ha garantito l'attuazione dei progetti e ti ha seguito e indirizzato verso le attività aziendali o di Project Work, mentre il secondo ti ha seguito durante l'attività in azienda, tenendo i contatti con il tutor scolastico.

I due tutor hanno progettato e verificato la tua attività di formazione fin dai primi momenti. In particolare, il tutor scolastico ti ha aiutato a valutare gli obiettivi raggiunti e le competenze da te progressivamente sviluppate, tenendo aggiornato costantemente il Consiglio di classe.

La certificazione delle competenze finale

Le **competenze certificate** sono crediti sia ai fini del percorso scolastico per il conseguimento del diploma o della qualifica, sia per gli eventuali passaggi tra i sistemi, compresa la transizione nei percorsi di apprendistato. Al termine di ogni esperienza svolta i tutor hanno elaborato insieme un report sui tuoi risultati. Il **Consiglio di classe** ha poi avuto il compito, alla fine di ogni anno, di certificare le competenze da te acquisite durante il percorso.

Il rapporto con l'azienda

Se hai svolto parte del percorso in ambiente lavorativo, non hai stabilito un rapporto di lavoro con l'azienda ospitante e, se minorenne, non hai acquisito la qualifica di «lavoratore minore». Sei stato però equiparato al lavoratore ai fini e agli effetti delle disposizioni dell'art. 20 del D. Lgs. 81/2008. Pertanto hai dovuto rispettare gli **obblighi propri di ciascun lavoratore**, e in particolare:

- hai avuto l'**obbligo di frequentare** e di attuare le attività formative concordate;
- hai dovuto rispettare gli orari concordati (e i due tutor hanno avuto l'**obbligo di controllare la tua frequenza**);
- hai avuto l'**obbligo di riservatezza** rispetto ai dati, alle informazioni e alle conoscenze relativi ai processi produttivi acquisiti durante lo svolgimento dell'attività formativa in azienda.

I principali tipi di attività

Le attività organizzate dalle scuole in collaborazione con le aziende possono essere state di vario tipo.

Le principali sono le seguenti:

- **seminari e workshop**;
- **visite aziendali**;
- **attività di stage** svolte nelle aziende;
- **i Project Work**, attività commissionate da aziende del territorio, da svolgere nei laboratori della scuola;
- **imprese simulate**;
- nel quinto anno anche giornate di formazione presso le **università**, per orientare anche verso la continuazione degli studi.

Le relazioni, il Portfolio e la presentazione dei risultati finali ottenuti

Per ogni attività normalmente ti è stato richiesto di stendere una relazione; oppure ti è stato somministrato un test. Entrambe queste attività fanno sicuramente parte del tuo **Portfolio**, utile nel seguito della tua carriera lavorativa. In alcuni casi sono anche state (o potranno essere) oggetto di valutazione.

Di solito alla fine dell'anno scolastico le scuole organizzano un momento di condivisione dell'attività svolta, cui partecipano gli studenti, i docenti, i rappresentanti delle aziende e le famiglie. Durante questi momenti gli studenti presentano i risultati finali ottenuti. È un momento utile, che insegna a rielaborare le esperienze fatte e a presentarle, in maniera sintetica ma completa, a un vasto pubblico. È sicuramente un esercizio che prepara anche per il colloquio orale dell'esame e per i futuri colloqui di lavoro.

Il Diario di bordo

Si tratta di un diario in cui tener traccia giorno per giorno delle attività svolte, dei problemi sorti, delle soluzioni ipotizzate e di quelle scelte. Il Diario di bordo è utilizzato sia dai tutor sia dagli studenti impegnati nelle attività formative ed è uno strumento essenziale per la valutazione.

L'attività più frequente: lo stage

La più tipica metodologia per acquisire competenze trasversali e professionali è lo stage svolto nell'azienda. L'organizzazione dello stage prevede che la scuola prenda contatto con le aziende del territorio, stabilisca accordi con quelle disponibili, progetti in collaborazione con queste ultime percorsi formativi personalizzati e decida il periodo della loro realizzazione.

Dopo aver individuato i profili richiesti da ogni azienda, si è provveduto all'abbinamento tra una azienda e un allievo in base ai dati acquisiti nelle fasi precedenti, con lo scopo di individuare la giusta azienda per ogni studente. A volte le aziende chiedono di ricevere una presentazione dell'allievo, che poi convocano per un colloquio diretto. È stato importante quindi saper preparare un **curriculum vitae**, un'abilità che ti servirà sicuramente anche una volta diplomato.

Un errore che spesso viene commesso nella compilazione del curriculum vitae è quello di sottovalutare

le proprie esperienze di vita.

Alla tua età le **competenze professionali**, basate sull'esecuzione di compiti specifici, saranno sicuramente poche. Quindi è necessario che tu metta in mostra le tue attitudini generali e le tue **competenze trasversali**, cioè quelle legate alle tue capacità sociali ed emotive e alle tue attitudini personali, come – per esempio – la capacità di concentrarti sugli obiettivi o quella di saperti organizzare.

Per esempio, se fai sport e giochi in una squadra (calcio, basket, volley), evidenzialo: è sintomo che sarai in grado di lavorare in gruppo e se magari sei anche il capitano, evidenzi doti da leader. Se sei animatore in oratorio, indicalo: vorrà dire che sei in grado di organizzare attività, di gestire persone e di rapportarti con piccoli e adulti (i genitori). Se hai un hobby, mostralo: vuol dire che hai molteplici interessi.

La metodologia del Project Work

I percorsi per le competenze trasversali e l'orientamento possono prevedere anche l'utilizzo di altre metodologie, in alternativa o in affiancamento allo stage: la più importante è il Project Work.

In particolare la metodologia del Project Work prevede che un'azienda proponga alla scuola un progetto, che verrà discusso con il referente di specializzazione al fine di renderlo effettivamente realizzabile nei laboratori della scuola. Vengono quindi individuati un gruppo di studenti (in genere, una classe) e un tutor scolastico che seguirà il progetto. L'obiettivo è realizzare un prodotto finale, rispettando vincoli temporali e tecnologie aziendali. Nei laboratori il tutor aziendale è presente e opera in sinergia con il tutor scolastico. Gli studenti svolgono compiti e risolvono problemi nati da fabbisogni reali, operando in un contesto collaborativo. In tale contesto sviluppano anche le competenze relazionali che hanno un'importanza primaria nel mondo del lavoro.

■ IL CURRICULUM VITAE

Un documento molto importante per chi vuole mettersi in contatto con il mondo del lavoro è il curriculum vitae. Il curriculum vitae espone in modo chiaro e sintetico le esperienze personali, scolastiche e lavorative di chi lo redige.

Spesso il curriculum vitae è la principale fonte informativa in base alla quale un datore di lavoro o un selezionatore decide se è interessato a esaminare ulteriormente un candidato per valutarne l'assunzione o la collaborazione. Pertanto deve essere redatto con cura, al fine di evitare errori e di renderlo di facile e veloce consultazione.

Vi sono molte tipologie di curriculum facilmente reperibili in Internet. Uno dei più utilizzati e richiesti è quello nel Formato Europeo.

FORMATO EUROPEO PER IL CURRICULUM VITAE



INFORMAZIONI PERSONALI

Nome

Indirizzo

Telefono

Fax

[Cognome, nome e, se pertinente, altri nomi.]

[Numero civico, strada o piazza, codice postale, città, Paese.]

<p>IE-mail Nazionalità Data di nascita</p> <p>ESPERIENZA LAVORATIVA</p> <p>Date (da - a) Nome e indirizzo del datore di lavoro Tipo di azienda o settore Tipo di impiego Principali mansioni e responsabilità</p> <p>ISTRUZIONE E FORMAZIONE</p> <p>Date (da - a) Nome e tipo di istituto di istruzione o formazione Principali materie/abilità professionali oggetto dello studio Qualifica conseguita Livello nella classificazione nazionale (se pertinente)</p> <p>CAPACITÀ E COMPETENZE PERSONALI</p> <p><i>Acquisite nel corso della vita e della carriera ma non necessariamente riconosciute da certificati e diplomi ufficiali.</i></p> <p>MADRELINGUA ALTRE LINGUE Capacità di lettura Capacità di scrittura Capacità di espressione orale</p> <p>CAPACITÀ E COMPETENZE RELAZIONALI</p> <p><i>Vivere e lavorare con altre persone in ambiente multiculturale, occupando posti in cui la comunicazione è importante e in situazioni in cui è essenziale lavorare in squadra (per esempio, cultura e sport), ecc.</i></p> <p>CAPACITÀ E COMPETENZE ORGANIZZATIVE</p> <p><i>Per esempio coordinamento e amministrazione di persone, progetti, bilanci; sul posto di lavoro, in attività di volontariato (per esempio, cultura e sport), a casa, ecc.</i></p> <p>CAPACITÀ E COMPETENZE TECNICHE</p> <p><i>Con computer, attrezzature specifiche, macchinari, ecc.</i></p> <p>CAPACITÀ E COMPETENZE ARTISTICHE</p> <p><i>Musica, scrittura, disegno, ecc.</i></p> <p>PATENTE O PATENTI</p> <p>ULTERIORI INFORMAZIONI</p> <p>ALLEGATI</p>	<p>[Giorno, mese, anno.]</p> <p>[Iniziare con le informazioni più recenti ed elencare separatamente ciascun impiego pertinente ricoperto.]</p> <p>[Iniziare con le informazioni più recenti ed elencare separatamente ciascun corso pertinente frequentato con successo.]</p> <p>[Indicare la madrelingua.] [Indicare la lingua.] [Indicare il livello: eccellente, buono, elementare.] [Indicare il livello: eccellente, buono, elementare.] [Indicare il livello: eccellente, buono, elementare.]</p> <p>[Descrivere tali competenze e indicare dove sono state acquisite.]</p> <p>[Descrivere tali competenze e indicare dove sono state acquisite.]</p> <p>[Descrivere tali competenze e indicare dove sono state acquisite.]</p> <p>[Descrivere tali competenze e indicare dove sono state acquisite.]</p> <p>[Inserire ogni altra informazione pertinente: per esempio, persone di riferimento, referenze, ecc.]</p> <p>[Se del caso, enumerare gli allegati al CV.]</p>
---	--

2. RICONOSCERE E VALUTARE LE COMPETENZE TRASVERSALI

Lo scopo principale dei percorsi per le competenze trasversali e l'orientamento è quello di permetterti di rafforzare il possesso di conoscenze e competenze che hai acquisito a scuola e di svilupparne altre che ti siano utili in futuro nel mondo del lavoro.

Distinguiamo infatti le conoscenze e le competenze in due grandi categorie: quelle relative ai contenuti e alle procedure disciplinari, che sei da sempre abituato a utilizzare a scuola (**competenze disciplinari o verticali**), e quelle relative alle capacità relazionali, che sei molto meno abituato a utilizzare a scuola, ma che hanno un'importanza centrale nel mondo del lavoro (**competenze trasversali o relazionali**).

Nel mondo della scuola sono privilegiate le competenze relative alla disciplina studiata. Nel mondo del lavoro sono richieste non solo queste competenze, ma anche molte altre. Tu hai costruito e costrisci inconsapevolmente parte di queste competenze ulteriori mentre svolgi compiti scolastici.

Ne sviluppi alcune, però, in modo insufficiente; altre le trascuri quasi completamente.

Quali sono tali competenze ulteriori? Come avrai avuto modo di sperimentare ci sono, per esempio, la capacità di lavorare in gruppo, di comunicare con gli altri sapendo farsi capire, di coordinare il lavoro di altre persone, di utilizzare risorse presenti nell'organizzazione in cui si è inseriti, di rispettare i tempi di consegna del lavoro, di affrontare gli imprevisti, di concentrarsi sui compiti che devono essere svolti.

IL RAPPORTO CON IL MONDO DEL LAVORO E LA DIDATTICA PER COMPETENZE

Non solo le competenze disciplinari...

Orientati a sviluppare capacità spendibili nel mondo del lavoro, i percorsi di cui ci stiamo occupando ti permettono di applicare le competenze acquisite a scuola e di sviluppare nuove competenze.

... anche quelle di cittadinanza

Oltre alle competenze disciplinari o verticali, ti consentono di sviluppare anche quelle necessarie per collaborare in un gruppo di lavoro (saper collaborare con gli altri, saper comunicare, saper accettare critiche, saper rispettare i tempi di consegna), e, in generale, le **competenze di cittadinanza** (saper risolvere problemi, sapersi porre in un'ottica di continuo miglioramento, essere responsabili verso i compiti assegnati, saper acquisire e interpretare le informazioni).

Devi anche acquisire la capacità di riflettere su te stesso, sulla tua capacità di essere autonomo o meno, di saperti concentrare sui tuoi compiti, in generale di essere consapevole dei tuoi punti di forza e di debolezza.

■ LA VALUTAZIONE DELLE COMPETENZE DI CITTADINANZA

Data l'importanza che le competenze di cittadinanza rivestono nella valutazione, ti proponiamo un esempio di modello per valutarle.

La lettura dei livelli che puoi raggiungere chiarisce ulteriormente il significato delle competenze trasversali e ti permette di provare ad autovalutarti.

Esempio di modello di valutazione delle competenze di cittadinanza

COMPETENZE DI CITTADINANZA (competenze trasversali)	L.I. (insufficiente/ non adeguato)	L.B. (base/ principiante)	L.M. (medio/ praticante)	L.A. (alto/esperto)
IMPARARE A IMPARARE	Non è in grado di organizzare il proprio apprendimento in modo efficace né dal punto di vista teorico, né da quello pratico.	Non sempre organizza il proprio apprendimento in modo efficace (tempi, fonti, risorse, tecnologia) con adeguati risultati teorico-pratici.	Organizza il proprio apprendimento in modo efficace (tempi, fonti, risorse, tecnologia) con discreti risultati teorico-pratici.	Organizza e pianifica il proprio apprendimento (tempi, fonti, risorse, tecnologia) con ottimi risultati teorico-pratici.
PROGETTARE, ELABORARE E REALIZZARE PROGETTI	Non partecipa al processo di ideazione, mostrando disinteresse nel formulare proposte alternative o nell'utilizzare le proprie conoscenze.	Partecipa superficialmente al processo di ideazione, non riuscendo sempre a formulare proposte alternative completamente valide, anche se accettabili.	Partecipa al processo di ideazione, formulando proposte alternative e utilizzando con sicurezza le proprie conoscenze.	Partecipa attivamente al processo di ideazione, formulando alternative, utilizzando con sicurezza le proprie conoscenze, valutando e definendo le strategie migliorative.
COMUNICARE, COMPRENDERE E RAPPRESENTARE	La capacità espressiva è globalmente scorretta, sia nella sintassi sia nel lessico, comprendendo e rappresentando il messaggio in modo inadeguato con linguaggi e supporti diversi.	La capacità espressiva è globalmente corretta, sia nella sintassi sia nel lessico, comprendendo e rappresentando il messaggio in modo semplice e accettabile con linguaggi e supporti diversi.	La capacità espressiva è efficace e discretamente corretta, sia nella sintassi sia nel lessico, comprendendo e rappresentando il messaggio con una discreta proprietà di linguaggi e supporti diversi.	La capacità espressiva è efficace e completamente corretta, sia nella sintassi sia nel lessico, comprendendo e rappresentando il messaggio con disinvolta, nei linguaggi e con supporti diversi.
COLLABORARE E PARTECIPARE	Si rifiuta di partecipare e interagire in gruppo in modo propositivo e non offre mai la propria disponibilità d'aiuto e di collaborazione.	Non sempre riesce a partecipare e a interagire in gruppo in modo propositivo e solo saltuariamente offre la propria disponibilità d'aiuto e di collaborazione.	Riesce a partecipare e a interagire in gruppo in modo propositivo, offrendo la propria disponibilità d'aiuto e di collaborazione.	Riesce a partecipare e interagire in gruppo con entusiasmo in modo attivo e propositivo, offrendo disponibilità d'aiuto e di collaborazione.

COMPETENZE DI CITTADINANZA (competenze trasversali)	L.I. (insufficiente/ non adeguato)	L.B. (base/ principiante)	L.M. (medio/ praticante)	L.A. (alto/esperto)
AGIRE IN MODO AUTONOMO E RESPONSABILE	Non è autonomo, consapevole, rispettoso dei propri doveri e dei diritti altrui, fatica a riconoscere le norme e non assolve il lavoro assegnato.	È sufficientemente autonomo, consapevole, rispettoso dei propri doveri e dei diritti altrui, riconoscendo le norme e assolvendo, pur faticosamente, il lavoro assegnato con diligenza e puntualità.	È discretamente autonomo, consapevole, rispettoso dei propri doveri e dei diritti altrui, riconoscendo le norme e assolvendo il lavoro assegnato con diligenza e puntualità.	È completamente autonomo, consapevole, rispettoso dei propri doveri e dei diritti altrui, riconoscendo le norme e assolvendo il lavoro assegnato con diligenza, grande precisione e puntualità.
RISOLVERE PROBLEMI (capacità di risoluzione, sguardo laterale e capacità di mediazione)	Non riesce a individuare soluzioni alternative e risolutive a situazioni critiche impreviste, rifiutandosi di usare intuito e logica.	Individua adeguate soluzioni alternative e risolutive a situazioni critiche impreviste, usando intuito e logica.	È in grado di individuare discrete soluzioni alternative e risolutive a situazioni critiche impreviste, usando con efficacia intuito e logica.	È in grado di individuare brillantemente soluzioni alternative a situazioni critiche impreviste e/o conflittuali, usando con efficacia logica, intuito e buon senso.
INDIVIDUARE COLLEGAMENTI E RELAZIONI	Non comprende e riconosce le modalità, le cause e gli effetti di eventi, dati, procedure e progressi, non individuandone le continuità e le discontinuità.	A fatica comprende e riconosce le modalità, le cause e gli effetti di eventi, dati, procedure e processi, non sempre individuandone le continuità e le discontinuità.	Comprende e riconosce le modalità, le cause e gli effetti di eventi, dati, procedure e processi, individuandone le continuità e le discontinuità.	Comprende e riconosce le modalità, le cause e gli effetti di eventi, dati, procedure e processi, individuandone le continuità e le discontinuità in modo originale.
ACQUISIRE E INTERPRETARE L'INFORMAZIONE	Non è in grado di ricercare, raccogliere e rielaborare dati e informazioni, non riuscendo a motivare le proprie opinioni.	Ricerca, raccoglie e rielabora dati e informazioni, motivando adeguatamente le proprie opinioni.	Ricerca, raccoglie e rielabora documenti, dati e informazioni con attenzione, motivando discretamente le proprie opinioni.	Ricerca, raccoglie e rielabora documenti, dati e informazioni con attenzione ed efficacia, motivando le proprie opinioni in modo convincente e originale.

3. ESEMPIO DI UNA RELAZIONE CONCLUSIVA SUI "PERCORSI"

Nel seguito mostriamo un esempio di relazione conclusiva sulle attività per le competenze trasversali e per l'orientamento. Per ogni parte della relazione verranno dati consigli su come realizzarla; è invece a disposizione sul sito la relazione completa. L'esempio si basa su una esperienza realmente effettuata da studenti come te negli anni scorsi.

■ Intestazione, prima pagina e sommario

Può sembrare banale, ma una prima pagina ben fatta permette già a chi dovrà leggere la relazione di intuirne il contenuto e di farsi un'idea del lavoro. Pertanto la prima pagina deve presentare tutte le informazioni necessarie, ma esposte in modo accattivante, per convincere il lettore ad aprire la relazione e a leggerla.

Dovrà quindi contenere nome, cognome, classe dello studente, nonché i riferimenti per identificare la scuola. Il titolo dovrà essere ben evidenziato e magari dovrà essere accompagnato da un sottotitolo più esplicativo. Poiché un'immagine vale più di mille parole, cercate un'immagine significativa per rendere meno "asciutta" la prima pagina.

Il sommario è di fatto l'indice della vostra relazione e permette a tutti di vedere come si è sviluppato il lavoro e a ognuno di decidere, volendo, di vedere direttamente solo le parti di suo interesse. Tramite il word processor che utilizzi puoi creare l'indice in modo automatico, così che rimanga sempre aggiornato.

RELAZIONE SUI "PERCORSI PER LE COMPETENZE TRASVERSALI E PER L'ORIENTAMENTO"

Scuola:
Classe:
Sezione:
Studente:



SOMMARIO

5. Introduzione	1
6. Incontri e seminari	1
7. Contatti con le aziende	2
8. Il Project Work	2
9. L'attività in azienda (stage)	2
10. Il prodotto finale	3
11. Ringraziamenti	4
12. Conclusioni personali	5
13. Autovalutazione	5

■ Introduzione

Nella introduzione andrà descritto in modo sintetico il percorso svolto nei tre anni con l'indicazione delle tipologie di attività realizzate, che verranno poi approfondite nei paragrafi successivi della relazione. In particolare andrà indicata la suddivisione oraria effettuata e il fatto di avere partecipato a dei corsi (sulla sicurezza, sulle soft skill e di tipo tecnico). Andranno poi indicate le tipologie di attività svolte con le aziende.

1. Introduzione

Il progetto si è svolto durante gli ultimi tre anni di corso e ha portato ad acquisire competenze trasversali e disciplinari attraverso momenti seminari e di attività in cooperazione con le imprese. In particolare questo è avvenuto tramite alcune tipologie di attività:

- incontri con persone provenienti dal mondo aziendale e seminari/corsi mirati all'acquisizione di una specifica competenza;
- un Project Work svolto a scuola in collaborazione con una azienda esterna;
- un periodo di stage presso un'azienda informatica;
- un progetto finale svolto con un'azienda mirato alla realizzazione di un prodotto.

■ Descrizione delle attività

In questa parte andranno descritte tutte le attività svolte nei tre anni, suddivise o in base all'anno, o in base alla tipologia. Quest'ultima modalità è preferibile, in quanto permette di capire meglio che cosa si è veramente fatto e che tipi di competenze si sono acquisite.

In particolare vi sono due grosse categorie di attività:

- formazione/informazione;
- attività dirette con le aziende.

Pertanto nella relazione, sotto il titolo *Incontri e seminari*, dovremo innanzitutto descrivere la formazione, mettendo in evidenza tutte le attività svolte, suddivise in:

- formazione sulla sicurezza;
- formazione su competenze trasversali (soft skill);
- formazione tecnica mirata al lavoro in azienda;

2. Incontri e seminari

Tra i corsi/seminari seguiti più significativi possiamo indicare i seguenti.

- Corso sulla sicurezza: abbiamo seguito un corso base di 4 ore sulla sicurezza, in cui abbiamo affrontato gli aspetti normativi e il concetto di rischio. Successivamente abbiamo seguito un corso specifico di otto ore, in cui abbiamo capito i rischi e le modalità di prevenzione legate all'uso di apparecchiature informatiche.
- Soft & Problem Skills: c'è stata una introduzione alle strategie di Problem Solving con l'illustrazione delle loro fasi:
 - a. comprensione del problema da risolvere, basata su 3 componenti: il contesto, le complicazioni e le domande chiave;
 - b. pensiero creativo e metodi innovativi e diversi per la risoluzione del problema. Questa strategia può essere approcciata tramite due tecniche opposte: metodo deduttivo (Top-Down), che

- partendo dal generale arriva al particolare, e metodo induttivo (Bottom-Up), che partendo da singoli casi particolari cerca di stabilire una legge universale;
- c. destrutturazione del problema in questioni chiave per l'analisi singolare.
- Simulazione, colloquio e redazione del Curriculum Vitae: le lezioni hanno avuto l'intento di prepararci a futuri colloqui di lavoro (singoli e di gruppo) e di insegnarci come scrivere al meglio un Curriculum Vitae, in modo da distinguerci dai competitori.
- Dopo un'accurata esposizione introduttiva, abbiamo simulato un colloquio di lavoro con alcuni *recruiter* aziendali (cioè addetti alla ricerca e alla selezione del personale).
- La fase successiva ha riguardato come svolgere una corretta scrittura di un Curriculum Vitae, sottolineandone tutti gli aspetti, dalle caratteristiche della forma (breve, chiara, corretta, personalizzata) a quelle della sostanza (anagrafica, formazione, esperienza lavorativa, competenze, interessi).
- Management by objectives: le lezioni hanno avuto l'intento di spiegare tutto ciò che riguarda un progetto, la sua definizione, il project management, il project management office e i suoi livelli e le fasi di progetto. Questo ci ha permesso di creare un esempio di progetto, completo di fasi, risorse e dei vari diagrammi (WBS, OBS, Gantt).

Altre competenze sono state acquisite tramite incontri/testimonianze con personaggi del mondo industriale. In particolare abbiamo incontrato:

1. l'amministratore delegato di una piccola software house, che ci ha spiegato che cosa è un'impresa e cosa vuol dire essere imprenditori;
2. un manager di una grossa multinazionale, che ci ha fatto comprendere come sono organizzate le grosse aziende e come si lavora al loro interno;
3. un ex allievo della nostra scuola, che ci ha mostrato la sua start-up e come abbia deciso di mettersi in gioco fin da subito per cercare di concretizzare i suoi sogni;
4. un formatore, che ha spiegato che cosa vuol dire parlare in pubblico, come comportarsi sia quando si deve illustrare un proprio progetto a un gruppo di persone, sia quando si partecipa a una riunione. Con lui abbiamo anche simulato le situazioni spiegate.

In seguito, sotto i titoli *Contatti con le aziende*, *Il Project Work* e *L'attività in azienda (stage)*, dovremo descrivere le attività effettivamente svolte che, nei vari anni, possono venire declinate in modo differente. In particolare:

- *Project Work* (in genere effettuati in terza);
- *stage* (svolti prevalentemente in quarta e in quinta);
- *realizzazione di prodotti* (svolti in quinta e molto spesso come continuazione del lavoro effettuato durante lo stage).

3. Contatti con le aziende

Questa fase è consistita, prima nella visita di un'azienda, poi nella presentazione delle aziende disponibili per gli stage. La visita all'azienda TILAB (Telecom Italia Lab) è avvenuta durante il terzo anno. Una guida ha presentato i 3 laboratori di ricerca: WIN (Wireless Innovation), Open Air (smart city) e IN (Innovation) Labs.

Un'altra opportunità di scoprire il tessuto imprenditoriale nel nostro territorio è stata la presentazione delle aziende durante il quarto anno. I rappresentanti delle aziende sono venuti a scuola, si sono presentati, hanno proposto un progetto per lo stage. Chi era interessato, poteva proporsi per un colloquio.

4. Il Project Work

Il Project Work, avvenuto in terza, è stato svolto a scuola con la collaborazione di un'azienda esterna: Raneri Web Design. L'obiettivo del progetto era la realizzazione di un sito web per pubblicizzare l'azienda. Questa richiesta è stata gestita dividendo la classe in due gruppi: il primo si sarebbe occupato della forma e il secondo del contenuto. Il primo si è occupato principalmente dell'aspetto grafico e io ne ero il *team leader*. Gli strumenti utilizzati sono stati WordPress, un software CMS (Content Management System), e FileZilla, un software per il trasferimento di file.

5. L'attività in azienda (stage)

L'azienda presso cui ho fatto lo stage è una società che opera dal 1995 nei settori della Consulenza Organizzativa, Information Technology, Ingegneria di Progetto e Prodotto, Formazione Aziendale.

L'azienda è stata costituita allo scopo di fornire alle imprese un supporto gestionale e operativo durante il processo di creazione di programmi di innovazione tecnologica e di sviluppo di prodotti e servizi. La mia principale occupazione durante il periodo di stage, dopo un'iniziale fase di formazione, è stata la migrazione di dati tra due differenti CMS: WordPress e Joomla!

Un CMS (Content Management System) è un software applicativo, installato su un server web, che permette di facilitare la gestione dei contenuti di un sito.

Le operazioni effettuate sul sito sono state:

- migrazione dati da WordPress a Joomla!;
- modifica del template del sito;
- traduzione di articoli (da italiano a inglese);
- inserimento dei moduli necessari, per esempio slideshow, e-commerce, login e scelta della lingua.

Tra gli aspetti più positivi dell'esperienza è che ho imparato a lavorare efficientemente in gruppo. Essendo infatti coinvolte nel progetto più persone, è stato necessario suddividere i compiti in maniera equa in base alle capacità e competenze di ognuno.

Ho avuto inoltre l'opportunità di relazionarmi con persone con molta esperienza nel settore dell'informatica, che hanno saputo consigliarmi efficacemente per la risoluzione dei problemi riscontrati. È stata un'esperienza aziendale a tutti gli effetti, che mi ha formato sotto il punto di vista lavorativo e personale, rendendomi cosciente di ciò che mi aspetterà una volta finite le scuole superiori.

Le uniche note "dolenti" sono state l'orario di lavoro non troppo flessibile e il non poter sempre mettere in pratica ciò che ho imparato durante i corsi di formazione che si sono tenuti in azienda, che spaziavano dalla gestione di un database alla creazione di applicazioni per dispositivi Android, dato che l'attività di stage non comprendeva questi progetti. Sicuramente gli apprendimenti mi saranno utili nel corso di attività future.

■ IL PRODOTTO FINALE

La realizzazione di prodotti finali è svolta prevalentemente in quinta e molto spesso come continuazione del lavoro effettuato durante lo stage. La descrizione del prodotto finale deve essere molto sintetica, poiché sarebbe meglio che fosse oggetto di una specifica relazione mirata alla sua presentazione orale.

6. Il prodotto finale

Durante l'ultimo anno abbiamo consolidato le competenze acquisite, mettendole in pratica nella realizzazione di un prodotto conclusivo. Nel mio caso questo fatto è avvenuto in collaborazione con un'azienda di sviluppo software.

Sono stati fatti una serie di incontri per consolidare o acquisire alcune competenze di base per affrontare in modo consapevole il mondo del lavoro. Alcune di queste tematiche erano già state affrontate negli anni precedenti, ma in questa fase, ormai prossimi alla fine del ciclo di studi, e con alle spalle già due anni di esperienza, sono state viste in una nuova ottica e con una nostra maggiore consapevolezza.

Ho potuto apprendere metodologie specifiche per lo sviluppo di progetti, che mi hanno permesso la realizzazione di un prodotto "vero".

Nel mio caso, seguiti dai tutor aziendali, abbiamo prodotto un'applicazione mobile sfruttando la tecnologia Bluetooth. È stata realizzata una applicazione che implementasse il gioco del Tris su uno smartphone.

Nel farlo abbiamo utilizzato Android Studio, che è l'ambiente di sviluppo ufficiale per applicazioni Android sviluppato da Google stessa per il proprio sistema operativo. La API, l'insieme delle procedure da apprendere, è molto ben documentata e Android Studio ha una vastissima base di utenti; ciò ha notevolmente facilitato l'apprendimento, potendo attingere sia dalla documentazione ufficiale che dai forum.

■ Ringraziamenti

Nei ringraziamenti vanno indicate persone e aziende che si ritiene abbiano svolto un ruolo attivo e significativo nel proprio percorso.

7. Ringraziamenti

Speciali ringraziamenti vanno a tutti i collaboratori dell'azienda per l'impegno e l'accoglienza offertaci e ai nostri professori, che hanno supervisionato l'interazione tra l'istituto e l'azienda, rendendo possibili gli incontri.

In particolare ringraziamo le persone citate nella tabella.

Nominativo	Ruolo ricoperto
Prof. Rossi	Referente ASL PCTO classe III
Prof.ssa Bianchi	Referente ASL PCTO classe IV
Prof.ssa Verdi	Referente ASL PCTO classe V
Prof. Romani	Tutor scolastico classe IV
Ing. Luisi	Tutor aziendale e Referente aziendale BCD S.R.L, specializzata nella produzione software e adibita al coinvolgimento degli studenti nella fase specialistica del progetto
Dr. Roventi	Direttore Commerciale BCD S.R.L, principale referente dell'azienda. La sua presenza è stata fondamentale e costante durante lo svolgimento dell'esperienza.

■ Conclusioni personali e autovalutazione

Durante i 3 anni ti è stato spesso chiesto di dare delle valutazioni su te stesso, sull'attività, sull'azienda, sull'esperienza in generale svolta. Queste valutazioni possono essere state richieste in modo "libero" o attraverso dei questionari specifici. In genere ogni scuola predispone differenti moduli e non sempre ogni cosa è oggetto di valutazione. Una valutazione seria e sincera, oltre a essere sintomo di maturità, ti aiuta sicuramente a capire se tutto ha funzionato e se vi è qualche punto a cui dovrai prestare più attenzione nel futuro.

Dovrai quindi fornire le tue considerazioni sull'esperienza triennale, evidenziando gli aspetti positivi e quelli che potrebbero essere migliorati. Potrai anche inserire alcuni dei questionari da te compilati.

8. Conclusioni personali

In conclusione, il progetto è riuscito nel suo intento di far acquisire agli alunni capacità lavorative sia generali che specifiche. Infatti tali capacità hanno permesso di realizzare un prodotto funzionante in tre sole giornate lavorative, in modo quasi completamente autonomo.

9. Autovalutazione

QUESTIONARIO DI AUTOVALUTAZIONE DOPO L'ESPERIENZA TRIENNALE

1. Relativamente al periodo di attività triennale, sono stato valutato in termini di:

	Insuff.	Suff.	Discr.	Buono	Ottimo
Attenzione prestata al tutor				X	
Comprensione delle comunicazioni/spiegazioni					X
Partecipazione e coinvolgimento nella vita aziendale				X	
Responsabilità dimostrata nell'esecuzione del lavoro				X	
Puntualità e rispetto dell'orario di lavoro			X		
Grado di autonomia nell'eseguire il lavoro				X	
Disponibilità a riconoscere gli errori e correggersi					X
Rispetto del materiale affidato				X	
Propensione a eseguire i compiti rispettando i tempi assegnati				X	
Inserimento nell'ambiente di lavoro e nei rapporti interpersonali					X
Collaborazione e disponibilità ad aiutare gli altri					X
Disponibilità ad affrontare problematiche e compiti nuovi				X	

2. Ti consideri soddisfatto di aver partecipato ai percorsi per le competenze trasversali e per l'orientamento?

- A Poco
 B Abbastanza
 C Molto

3. Ritieni comunque di aver tratto vantaggio da questa esperienza?

- A Poco
 B Abbastanza
 C Molto

4. Con questa esperienza credi di avere avuto la possibilità di:

	Poco	Abbastanza	Molto
Conoscere il mondo del lavoro nel suo complesso		X	
Comprendere il settore in cui si colloca il tuo percorso formativo			X
Conoscere l'ambiente, le tecniche, l'organizzazione del lavoro			X
Accrescere le tue conoscenze professionali		X	
Sapere se nel luogo di lavoro frequentato è possibile realizzarsi o meno	X		
Aumentare la probabilità di trovare un posto di lavoro al termine dei tuoi studi		X	

QUESTIONARIO SULL' ATTIVITÀ PRESSO L'AZIENDA

1. L'esperienza pratica presso le aziende (stage) è stata:

- A Poco proficua
- B Abbastanza proficua
- C Molto proficua

2. Ritieni che il periodo di attività formativa debba avere una maggiore durata?

- A Sì
- B No
- C È sufficiente quello proposto

3. Valuta i seguenti aspetti della tua esperienza presso l'azienda in cui hai effettuato lo stage.

	Nullo	Basso	Medio	Alto
Relativamente ai compiti che ti sono stati assegnati, indica il grado di difficoltà incontrato			X	
Quanto impegno ti sembra di avere messo nel lavoro svolto				X
Come ti è sembrata la qualità dei rapporti con gli altri colleghi			X	
Sei riuscito ad acquisire gli elementi conoscitivi e di competenza previsti dal tuo progetto			X	
Hai compreso le informazioni e le hai utilizzate in modo coerente nell'affrontare i compiti assegnati			X	
Hai chiesto aiuto nel momento delle difficoltà		X		
Hai saputo cogliere gli imprevisti, partecipando alla loro soluzione			X	
Sei stato in grado di comunicare in modo efficace in forma scritta, usando gli strumenti cartacei e informatici				X
Sei soddisfatto dell'immagine di te che hai trasmesso			X	
Il tuo livello complessivo di soddisfazione di questa esperienza è				X

Relazione e presentazione di un progetto per il colloquio orale dell'esame di Stato

Lezione 1

Redigere la relazione di un
progetto: un esempio

Lezione 2

Presentare un progetto: un esempio

1. REDIGERE LA RELAZIONE DI UN PROGETTO: UN ESEMPIO

Negli istituti tecnici è abbastanza normale presentare al colloquio orale un lavoro particolare fatto durante l'ultimo anno di studi. Può essere un prodotto realizzato presso un'azienda durante uno stage compiuto nell'ambito dei percorsi per le competenze trasversali e l'orientamento, oppure un lavoro interdisciplinare (magari anche interclasse) di realizzazione di un manufatto. Nel seguito mostriamo un esempio di relazione e un esempio di presentazione multimediale relativa alla progettazione di una Applicazione per ipovedenti per smartphone. Per ogni parte verranno dati consigli su come realizzarla e verranno poi messe a disposizione sul sito la relazione e la presentazione complete. Tale esempio è tratto da un'esperienza realmente effettuata da studenti come te.

■ Intestazione, prima pagina e sommario

La prima pagina della relazione deve contenere tutte le informazioni necessarie per convincere il lettore ad aprire la relazione e a leggerla. Visto che il progetto è stato realizzato anche in funzione del colloquio orale dell'esame di Stato, accanto al titolo metterai la scritta "Esame di Stato". Metterai poi un'immagine relativa al prodotto realizzato eseguendo il progetto e, accanto a essa, l'indicazione della scuola in cui lo hai realizzato e il tuo nome.

Molto utile nella prima pagina è anche un piccolo abstract (riassunto), per far sapere subito a chi legge qual è il contenuto della relazione.

RELAZIONE SU: APP OCR ANDROID - ESAME DI STATO



Scuola:

Classe:

Sezione:

Studente: Stefano Gioda

Abstract

Il mio progetto si pone l'obiettivo di aiutare le persone con disturbi visivi nella lettura di documenti stampati, visto che il numero di individui affetti è in continua crescita.

La mia idea è di realizzare un'applicazione OCR (Optical Character Recognition) per smartphone Android, cioè un'applicazione in grado di convertire il testo di un'immagine in caratteri comprensibili dal computer. Successivamente le parole vengono dalla macchina o mostrate ingrandite sullo schermo oppure lette ad alta voce.

In questa prima versione l'applicazione è in grado di riconoscere un solo carattere per volta all'interno dell'immagine; quindi il riconoscimento richiede del tempo, perché può avvenire solo dopo che il carattere ha occupato l'intero schermo; mentre la classificazione viene mostrata in tempo reale.

L'OCR è implementato utilizzando una rete neurale creata con la libreria TensorFlow e poi esportata nell'applicazione. L'applicazione è costruita con Android Studio.

La seconda pagina conterrà il sommario, che evidenzierà la struttura della relazione, come mostra l'esempio seguente.

SOMMARIO

1. Introduzione
2. Basi teoriche
 - 2.1. OCR
 - 2.2. Intelligenza artificiale, deep learning e reti neurali
3. Sviluppo
 - 3.1. Prima parte: OCR
 - 3.1.1. Strumenti utilizzati
 - 3.1.1.1. TensorFlow
 - 3.1.2. Fasi
 - 3.1.2.1. Preprocessing
 - 3.1.2.2. Training
 - 3.1.2.3. Testing
 - 3.2. Seconda parte: Android
 - 3.2.1. Strumenti utilizzati
 - 3.2.1.1. Android SDK e NDK
 - 3.2.1.2. Android Studio
 - 3.2.2. Applicazione
 4. Gestione
 - 4.1. Descrizione work packages
 - 4.2. Diagramma di Gantt
 5. Conclusioni
 6. Bibliografia/Sitografia
 7. Ringraziamenti

■ Introduzione

Nell'introduzione andrà descritto in modo sintetico lo scopo del lavoro, come è nata l'idea, le modalità di attuazione. Andranno descritti i risultati attesi e quelli effettivamente ottenuti.

1. INTRODUZIONE

Il mio progetto d'esame si pone l'obiettivo di aiutare le persone con disturbi visivi nella lettura.

Per le persone affette da disturbi gravi sono necessari prodotti specifici e costosi.

Tra quelli in commercio, la maggior parte consiste in apparecchi elettronici ad hoc, come penne o piccoli tablet, che ingrandiscono e/o convertono i caratteri scritti.

La mia idea è quella di convertire i caratteri contenuti all'interno di un documento fisico in testo leggibile dal computer. Questo tipo di software viene chiamato OCR, Optical Character Recognition, cioè software "per il riconoscimento ottico dei caratteri".

Una volta effettuata la conversione, il testo può essere ingrandito o letto automaticamente dalla macchina. Per garantire la facilità d'uso, questa applicazione è sviluppata sulla piattaforma Android, dato che l'utilizzo dello smartphone è sempre più diffuso.

I prodotti in commercio, oltre a essere dispendiosi, possono essere ingombranti. La mia soluzione cerca di risolvere questi problemi; infatti verrà rilasciata gratuitamente e non richiede l'uso di dispositivi particolari; necessita solo di uno smartphone, che è notevolmente comodo e portatile.

In questo documento viene presentata la prima versione dell'applicazione Android, che è in grado di riconoscere un carattere alla volta all'interno di un'immagine.

■ Basi teoriche

In questa parte andranno descritte tutte quelle informazioni di base necessarie per comprendere il lavoro presentato.

Dovranno essere semplici, affinché le capisca anche chi non è un tecnico, ma precise e corrette, per mostrare le proprie competenze relative all'argomento in oggetto. Può essere utile dare alcune nozioni di base iniziali per tutti e dedicare poi un sottoparagrafo più dettagliato per chi voglia approfondire. È meglio se poi il tutto viene corredata da immagini e schemi.

2. BASI TEORICHE

L'applicazione si compone di due parti: applicazione Android e OCR, che si basa pesantemente su tecniche di intelligenza artificiale, deep learning e reti neurali.

2.1. OCR

Un OCR (Optical Character Recognition) è un sistema che permette di convertire i caratteri contenuti all'interno di un'immagine in testo digitale comprensibile dal computer ed è caratterizzato dalle seguenti fasi:

1. rilevazione del testo (*text detection*);
2. segmentazione dei caratteri (*character segmentation*);
3. classificazione dei caratteri (*character classification*).



fig. 1 Esempio di un OCR generico

2.2 Intelligenza artificiale, deep learning e reti neurali

L'OCR è realizzato utilizzando una rete neurale CNN che, dopo essere stata sviluppata, è esportata sul telefono. L'applicazione Android mostra in tempo reale la classificazione dell'immagine: infatti, per rendere più gradevole la *user experience*, non viene richiesto di scattare singole foto, ma il processo viene eseguito su un video in *real time*.

Tutte le immagini vengono passate alla rete neurale e i risultati vengono stampati a video. L'output è formato da un valore di confidenza per ogni carattere, che è un numero da zero a uno, e vengono mostrati i caratteri con l'output più alto. Dato che non è implementata la rilevazione del testo e la segmentazione, per il corretto funzionamento dell'applicazione è necessario inquadrare un carattere per volta, affinché occupi l'intera schermata.

■ Sviluppo

Questa è la parte principale del lavoro. Qui andranno indicate le fasi di sviluppo del progetto, le informazioni tecniche, gli ambienti di sviluppo, e tutto quello che può servire a descrivere ciò che si è fatto. In questa parte è bene essere precisi e rigorosi dal punto di vista tecnico, anche se non sempre è necessario dilungarsi eccessivamente con sigle e termini.

3. SVILUPPO

3.1 Prima parte: OCR

L'applicazione si compone di due parti:

1. riconoscimento dei caratteri (OCR);
2. applicazione Android.

In questa prima versione, nell'OCR è presente solo la conversione da immagine a carattere; invece la rilevazione del testo e la segmentazione sono ancora da implementare.

Se vengono usati strumenti, ambienti di sviluppo o prodotti commerciali è bene indicarli, dando eventualmente una breve descrizione.

3.1.1. Strumenti utilizzati

3.1.1.1. TensorFlow

TensorFlow è una libreria software inizialmente sviluppata da Google e, successivamente, rilasciata con licenza open source.

La versione principale utilizza il linguaggio di programmazione Python e fornisce moduli per il machine learning e per il deep learning.

Permette di definire staticamente la struttura di un modello di rete neurale dinamico nei valori dei neuroni. In particolare, è possibile definire delle variabili, alle quali non viene assegnato nessun valore, che formano l'input della rete neurale e sono chiamate *placeholders*.

Dopo aver addestrato la rete, quest'ultima può essere salvata e utilizzata per classificare le immagini fornite in input.

Un altro aspetto importante è che esiste una versione di TensorFlow per la piattaforma Android.

3.1.2. Fasi

3.1.2.1. Preprocessing

3.1.2.2. Training

3.1.2.3. Testing

Come è stato detto, il progetto è costituito da due sottoprogetti:

1. OCR;
2. applicazione Android.

La prima parte consiste dunque nella costruzione di un OCR mediante una rete neurale ed è a sua volta composta da alcune sottoattività:

1. preprocessing:
 - 1.1. ricerca del modello: scelta del modello di rete neurale più adatta al problema;
 - 1.2 ricerca del dataset, cioè delle immagini da usare per addestrare la rete neurale;
 - 1.3 manipolazione del dataset: operazioni di miglioramento e adattamento del dataset scelto;
2. training, addestramento della rete neurale;
3. testing, verifica dell'accuratezza ottenuta.

3.2. Seconda parte: Android

Il secondo sottoprogetto consiste nella creazione di un'applicazione Android che interfacci l'utente con la rete neurale ed è composto dalle seguenti sottoattività:

1. implementazione interfaccia grafica;
2. importazione della rete neurale addestrata.

Gestione

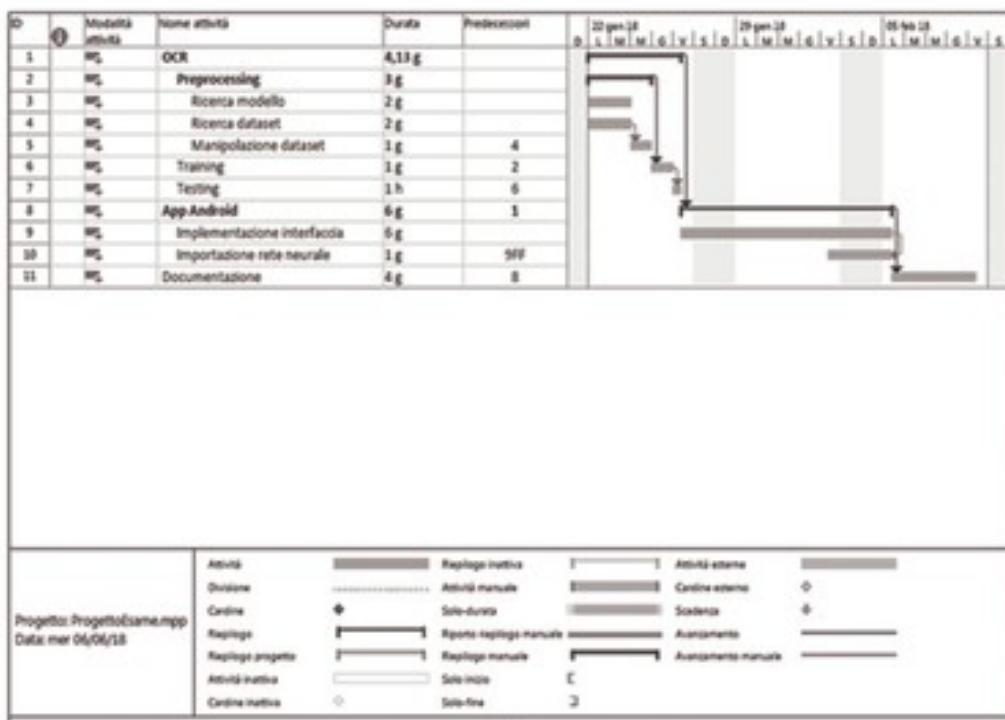
In questa fase va indicato come si è portato avanti il lavoro dal punto di vista gestionale, indicando quindi i vari Work Package, il legame tra essi e, da non sottovalutare, il diagramma di Gantt. In questo modo non solo si mostra la propria capacità di gestire i progetti, ma anche di avere recepito gli insegnamenti presenti in altre discipline (in questo caso GPOI).

4. GESTIONE

4.1. Descrizione work packages

4.2. Diagramma di Gantt

Per sviluppare il progetto ho provveduto preventivamente a programmare le attività per poter arrivare per tempo all'esame di Stato con il prodotto finito. Per far questo ho definito la WBS del progetto e pianificato le attività utilizzando il diagramma di Gantt.



Conclusioni

Nelle conclusioni si riassume brevemente lo scopo del lavoro e ciò che si è fatto. Vanno poi indicate le difficoltà incontrate, l'insegnamento appreso e i possibili sviluppi futuri.

Bibliografia/Sitografia

In questa parte andranno indicati i testi, le pubblicazioni e i siti che si sono utilizzati per lo svolgimento del proprio lavoro.

Ringraziamenti

Anche in questo caso, se si ritiene, vanno indicate, in modo molto sobrio, le persone che in qualche modo hanno permesso la realizzazione del lavoro.

5. CONCLUSIONI

Questa prima versione dell'applicazione è molto utile come dimostrazione della potenza delle reti neurali e del deep learning.

La parte cruciale

La parte cruciale, per poter realizzare l'applicazione, è stata la ricerca di un buon dataset di immagini per il training del modello. Fortunatamente, sono presenti numerosi dataset pubblici, in particolare quello scelto si è rivelato di buona qualità, perché contenente una quantità e una qualità di immagini adatta a rappresentare la realtà di riferimento.

Altre scelte importanti

Un'altra importante scelta è stata il tipo di rete neurale che si è rivelato adatto al tipo di problema; infatti ha fornito una discreta accuratezza e un'ottima velocità di esecuzione su un hardware limitato come quello di uno smartphone.

La libreria TensorFlow è stata di particolare aiuto, grazie alla sua documentazione esaustiva e ai numerosi strumenti messi a disposizione, che rendono molto semplice l'implementazione di una rete neurale.

Il completamento del progetto

Per completare il progetto è necessario realizzare il riconoscimento e la segmentazione del testo, usando un ulteriore modello di rete neurale.

Sviluppi futuri

Per quanto concerne l'applicazione Android, si potrà decidere se continuare a riconoscere il testo in tempo reale oppure richiedere all'utente di scattare una foto. Dopo aver convertito il documento, si potrebbe implementare la lettura automatica a voce alta, in modo da fornire un aiuto reale alle persone con disturbi visivi.

6. SITOGRAFIA

Android Studio: <https://tinyurl.com/ztauqmw>

Aumento disturbi visivi: <https://tinyurl.com/ybsq52a9>

Convolutional neural network: <https://tinyurl.com/yavzq3x3>

Dataset: <https://tinyurl.com/y9kr4gep>

MobileNet: <https://tinyurl.com/y9of5c2o>

MobileNet: <https://tinyurl.com/y6vr8xg3>

Repository: <https://tinyurl.com/q8767zh>

TensorFlow: <https://tinyurl.com/j5kw7zo>

7. RINGRAZIAMENTI

Speciali ringraziamenti vanno all'ing. Rossi della ditta ABC per gli utili suggerimenti tecnici e ai nostri professori, che hanno supervisionato lo svolgimento dell'attività.

2. PRESENTARE UN PROGETTO: UN ESEMPIO

■ Lo scopo della presentazione

Dopo aver stilato la relazione del progetto è consigliabile creare una presentazione con una decina di slide, che sarà di fondamentale aiuto durante l'esposizione orale. La presentazione serve:

- a te, per avere un riferimento per il tuo discorso, in modo da evitare scene mute; quindi la cosa più utile è inserire delle parole-chiave;
- agli ascoltatori, per seguire il filo del discorso; quindi deve essere accattivante ed esplicativa. Si può ottenere un effetto di questo tipo inserendo immagini (che spesso valgono più di mille parole).

■ Prima di iniziare

Il punto di partenza per la presentazione può essere la relazione, da cui devi scegliere le parti importanti, perché non avrai il tempo per raccontare tutto ciò che hai scritto. Una scelta sensata è quella di spiegare in modo particolarmente approfondito un pezzo del tuo progetto.

■ Il modello

I programmi di presentazione offrono svariati modelli: sceglie uno invitante, ma non complesso.

■ La sequenza delle diapositive

La presentazione non deve essere troppo lunga: dieci slide circa vanno bene. La struttura entro cui disporre in sequenza le diapositive è la seguente:

1. copertina;
2. introduzione;
3. svolgimento;
4. gestione del progetto;
5. conclusioni.

Ecco un esempio.

1. Copertina: conterrà il titolo del progetto, eventualmente un sottotitolo, i propri nome e cognome.



fig. 2 Copertina

- 2. Introduzione:** conterrà la presentazione del problema da cui si è partiti, specificando brevemente in che cosa consiste, e l'indicazione dei progetti/prodotti in commercio che possono fornirne la soluzione (1-2 diapositive).



fig. 3 Presentazione del problema



fig. 4 Presentazione dei prodotti in commercio che possono aiutare a risolvere il problema

- 3. Svolgimento:** conterrà la propria proposta di soluzione/sviluppo, prima descritta in termini generali, poi soffermandosi sulle singole componenti (specificando gli strumenti e le eventuali tecniche particolari utilizzate). Lo scopo è quello di mostrare sinteticamente tutto il processo di sviluppo (4-8 diapositive).



fig. 5 Presentazione della soluzione scomposta nelle diverse parti



fig. 6 Presentazione di una parte specifica della soluzione

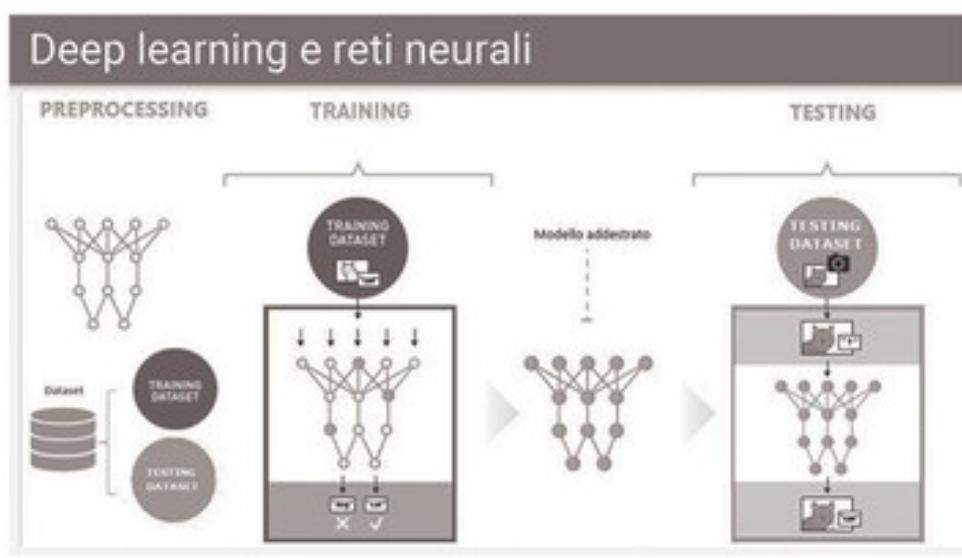


fig. 7 Presentazione di una tecnica specifica utilizzata



fig. 8 Presentazione degli strumenti utilizzati

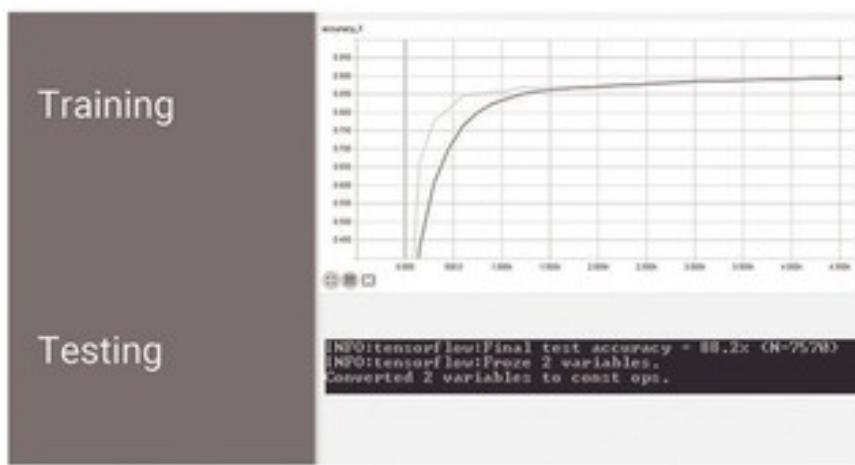


fig. 9 Presentazione del processo di sviluppo

4. **Gestione del progetto:** conterrà l'illustrazione delle metodologie di gestione del progetto utilizzate (per esempio, diagramma di Gantt, WBS – Work Breakdown Structure –, ...). (1-2 diapositive).

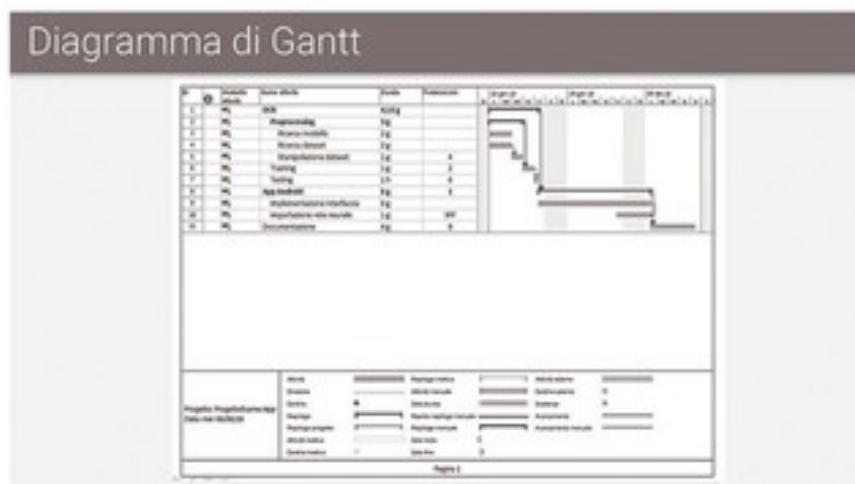


fig. 10 Presentazione del diagramma di Gantt

5. **Conclusioni:** conterrà la descrizione del risultato finale, messa a confronto con il problema e la proposta iniziale, e l'evidenziazione di eventuali difficoltà e di possibili sviluppi futuri (1-2 diapositive).

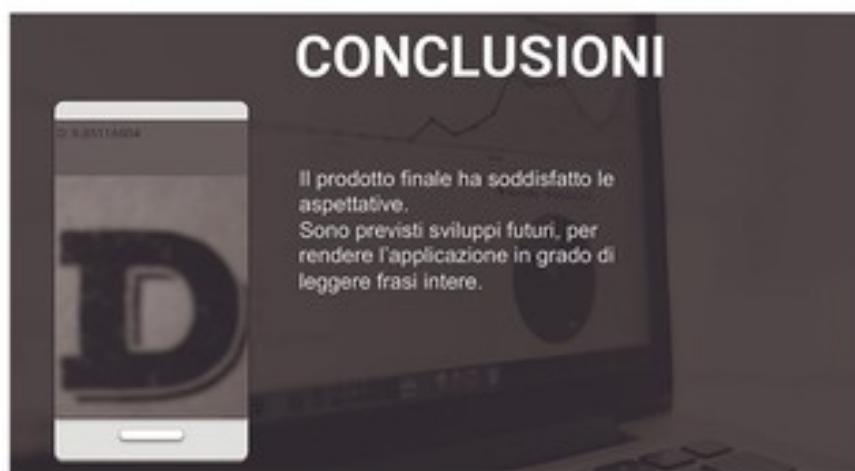


fig. 11 Presentazione del prodotto finale e delle conclusioni

Una volta che hai finito di stilare la presentazione non ti resta che farla vedere a tutti (in particolare ai tuoi professori), provando davanti a loro il discorso che vorrai fare al colloquio orale dell'esame.



PREPARAZIONE ALLA SECONDA PROVA SCRITTA DELL'ESAME DI STATO



Ti vengono ora forniti alcuni consigli per prepararti alla seconda prova scritta dell'esame di Stato. Nella prima sezione sono descritti tempi e modalità con cui si svolge la prova e viene illustrato come affrontarla. Ti sono forniti alcuni suggerimenti per comprendere la consegna e i punti su cui focalizzare l'attenzione durante lo svolgimento.

Nella seconda sezione ti viene proposto lo svolgimento completo di una prova d'esame, eseguito sul testo che hanno affrontato gli studenti dell'articolazione Informatica nell'anno scolastico 2016-2017.

La prova è costituita da due parti. La prima – che ha tipicamente a che vedere con la costruzione di un database – deve essere svolta per intero. La seconda è composta da una serie di quesiti a cui devi rispondere, scegliendo tra quelli proposti in base alle indicazioni fornite nella traccia. Qui, per completezza, troverai la risposta a tutti.

La prova fa riferimento a situazioni operative tipiche di un contesto tecnico professionale. Ti verrà chiesto di individuare le problematiche tecnologiche e organizzative coinvolte nel caso presentato e di proporre soluzioni progettuali. Le soluzioni potranno articolarsi in analisi, confronto, scelta, dimensionamento, sviluppo e implementazione, ottimizzazione, diagnostica, documentazione.

Nel caso in cui la prova assegnata concernesse più discipline, la traccia sarà predisposta, sia per la prima parte che per i quesiti, in modo da proporre temi, argomenti, situazioni problematiche che ti consentiranno di dimostrare, in modo integrato, le conoscenze, le abilità e le competenze da te acquisite nei diversi ambiti disciplinari.

*È possibile trovare tutte le tracce della seconda prova di informatica degli anni trascorsi sul sito del Ministero dell'Istruzione, dell'Università e della Ricerca, all'indirizzo:
<https://tinyurl.com/y7ulxjgu>.*



COME PREPARARSI AD AFFRONTARE LA SECONDA PROVA SCRITTA

È inutile dire che la preparazione all'esame inizia con lo studio durante l'anno e con lo svolgimento di esercizi come allenamento. È anche molto utile, per questo scopo, esercitarsi svolgendo i temi d'esame degli anni precedenti, misurando il tempo impiegato per lo svolgimento, in modo da saper organizzare il lavoro durante la prova. La durata della seconda prova per lo svolgimento del tema di informatica va **da sei a otto ore**. In genere viene consentito usare calcolatrici e manuali tecnici. Può quindi essere utile disporre di manuali per i linguaggi SQL, HTML e PHP o ASP.NET per la progettazione di pagine web.

I NUCLEI TEMATICI FONDAMENTALI E GLI OBIETTIVI DELLA PROVA

Ogni prova si svolge attorno ai **nuclei tematici fondamentali** e agli **obiettivi** elencati nella tabella seguente.

INFORMATICA	
Nuclei tematici fondamentali	Obiettivi della prova
<ul style="list-style-type: none">• Progettazione di basi di dati: modellazione concettuale, logica e fisica di una base di dati.• Sistemi gestionali di basi di dati: tipologie e funzionalità.• Linguaggi per basi di dati: creazione, manipolazione e interrogazione di una base di dati.• Tecnologie per il web: linguaggi lato client e lato server; realizzazione di applicazioni web anche con interfacciamento a basi di dati; principali aspetti di sicurezza delle applicazioni web.	<ul style="list-style-type: none">• Affrontare situazioni problematiche, utilizzando adeguate strategie cognitive e procedure operative orientate alla progettazione di soluzioni informatiche.• Sviluppare applicazioni e servizi informatici per reti locali o geografiche.• Scegliere sistemi e strumenti idonei al contesto proposto, in base alle loro caratteristiche funzionali.• Realizzare progetti secondo procedure consolidate e criteri di sicurezza.• Redigere relazioni tecniche e documentare le attività di progetto.

Dove potrebbero essere richiesti e come andrebbero affrontati gli obiettivi? Te lo indichiamo brevemente.

■ **Affrontare situazioni problematiche, utilizzando adeguate strategie cognitive e procedure operative orientate alla progettazione di soluzioni informatiche.**

Questo aspetto è quello che viene affrontato durante tutto il tema d'esame. In particolare nella prima parte, ma spesso anche in uno o più dei quesiti.

■ **Sviluppare applicazioni e servizi informatici per reti locali o geografiche.**

Il sistema da realizzare dovrà essere previsto per poter operare in rete. Quindi con applicazioni di tipo client-server. Di questo si dovrà tener conto nella soluzione del problema della prima parte, ma anche nelle eventuali richieste presenti nei quesiti.

■ **Scegliere sistemi e strumenti idonei al contesto proposto, in base alle loro caratteristiche funzionali.**

Normalmente nei quesiti della seconda parte possono venire proposte situazioni particolari che richiedono la scelta di specifici strumenti. Ma richieste analoghe possono essere presenti anche nella prima parte e può essere, comunque, utile accennarne nella proposta di soluzione.

■ **Realizzare progetti secondo procedure consolidate e criteri di sicurezza.**

Anche in questo caso possono esserci specifiche richieste tra i quesiti della seconda parte, ma è utile tenerne conto anche nella soluzione del sistema proposto al primo punto.

■ *Redigere relazioni tecniche e documentare le attività di progetto.*

Spesso uno dei quesiti riguarda la documentazione del sistema sviluppato nella prima parte.

Le varie parti della prova sono collegate primariamente a uno o ad alcuni dei **nuclei tematici fondamentali** e degli **obiettivi**. Come abbiamo già fatto nelle sezioni di preparazione all'esame di Stato collocate al termine delle Unità, nel corso dello svolgimento evidenzieremo tali collegamenti.

PRIMA PARTE DELLA PROVA

ANALISI DELLA CONSEGNA

Prima di iniziare a svolgere la prova d'esame è necessario leggere attentamente il testo della consegna, anche più volte, individuando i passaggi fondamentali e le richieste. Questo permetterà di capire la strada da seguire e organizzare un piano di lavoro che rispetti i tempi e risponda alle consegne.

Nella prova di informatica uno dei quesiti ricorrenti e centrali è quello di realizzare un'applicazione basata su un database.

Vediamo come procedere.

Nuclei tematici fondamentali	Obiettivi
<ul style="list-style-type: none"> Progettazione di basi di dati: modellazione concettuale, logica e fisica di una base di dati. 	<ul style="list-style-type: none"> Affrontare situazioni problematiche, utilizzando adeguate strategie cognitive e procedure operative orientate alla progettazione di soluzioni informatiche.

PROGETTAZIONE DI BASI DI DATI - Modellazione concettuale

EVIDENZIARE LE ENTITÀ, GLI ATTRIBUTI, LE RELAZIONI

Per progettare il database è importante evidenziare le entità del problema, gli attributi e le relazioni che emergono dal testo. Nel testo le entità di solito sono sostantivi, gli attributi sono aggettivi e le relazioni sono verbi.

Ricorda: l'ambito del problema non è mai una entità. Se per esempio si sta considerando la gestione di un istituto scolastico, sono entità gli Studenti, le Classi e i Docenti, ma non l'istituto, che non va considerato tra le entità.

Attenzione: un passaggio particolarmente importante è quello di riconoscere i vincoli imposti dalla consegna. Per esempio, se nella gestione degli iscritti a una palestra la consegna indica che questi devono avere un certificato medico con validità annuale, allora bisogna prevedere il relativo attributo nell'entità Iscritto.

COSTRUIRE IL MODELLO E/R

In secondo luogo, occorre costruire lo schema dei dati con le entità e gli attributi associati e validarlo, controllando che risponda a tutte le query SQL del problema. Inoltre bisogna descrivere i vincoli aggiuntivi imposti per semplificare il problema o chiarire i punti ambigui. Per esempio, se si richiede di differenziare gli animali di uno zoo in cuccioli o adulti, si può imporre di considerare tutti gli animali al di sotto di un anno come cuccioli e la rimanente parte come animali adulti.



Ricorda: *in questa fase è importante descrivere e motivare le scelte e i vincoli aggiuntivi che hanno portato alla creazione del modello di dati presentato.*

DOCUMENTARE IL MODELLO E/R

Una volta realizzato lo schema concettuale utilizzando il modello E/R, è necessario documentarlo, definendo i dizionari dei dati, delle associazioni e degli attributi.

RISTRUTTURARE IL MODELLO E/R

L'ultimo passaggio consiste nell'eliminare le eventuali gerarchie presenti e gli attributi multipli.

Modellazione logica

A questo punto, applicando le regole di derivazione, si traduce lo schema concettuale in **modello logico relazionale**, definendo le **tabelle interessate** e le **chiavi esterne**.

Una volta definito lo schema logico relazionale, bisogna verificare che sia normalizzato. Se non è in terza forma normale, bisogna normalizzarlo.

LINGUAGGI PER BASI DI DATI - Scrivere in linguaggio SQL le richieste al database

Nuclei tematici fondamentali	Obiettivi
<ul style="list-style-type: none">Linguaggi per basi di dati: creazione, manipolazione e interrogazione di una base di dati.	<ul style="list-style-type: none">Affrontare situazioni problematiche, utilizzando adeguate strategie cognitive e procedure operative orientate alla progettazione di soluzioni informatiche.

A questo punto si possono scrivere le richieste in SQL.

Attenzione: durante la scrittura delle richieste, se ci si accorge di non disporre dei dati necessari per risolvere una query, è necessario rivedere lo schema concettuale ed eventualmente modificarlo (sarà da modificare anche lo schema logico).

TECNOLOGIE PER IL WEB - Realizzazione del sito web

L'ultimo passo per terminare la prima parte della traccia sarà la codifica del progetto e la realizzazione del sito web.

Nuclei tematici fondamentali	Obiettivi
<ul style="list-style-type: none">Tecnologie per il web: linguaggi lato client e lato server; realizzazione di applicazioni web anche con interfacciamento a basi di dati; principali aspetti di sicurezza delle applicazioni web.	<ul style="list-style-type: none">Sviluppare applicazioni e servizi informatici per reti locali o geografiche.Realizzare progetti secondo procedure consolidate e criteri di sicurezza.

Ricorda: *a meno che non vi siano richieste specifiche, è preferibile scegliere una parte significativa del progetto per la codifica, meglio se è una parte che si interfaccia con il database, con lo scopo di dimostrare, anche solo con una piccola procedura, di saper scrivere applicazioni software.*



SECONDA PARTE DELLA PROVA

RISOLVERE I QUESITI

Da vari anni la struttura del testo d'esame prevede, oltre alla parte principale che abbiamo appena illustrato, una seconda parte composta da una serie di quesiti a cui devi rispondere, scegliendo tra quelli proposti in base alle indicazioni fornite nella traccia.

In alcune prove d'esame è richiesto di trattare un problema particolare come, per esempio, il problema della sicurezza dei dati. In questi casi bisogna fare riferimento alle conoscenze acquisite sull'argomento anche in altri ambiti o materie, per realizzare una trattazione completa.

Attenzione: è previsto che si risponda solo ad alcuni dei quesiti a scelta tra quelli proposti.

Ricorda: anche in questo caso, è importante leggere attentamente i testi di tutti i quesiti per scegliere quali risolvere.



ESEMPIO DI SVOLGIMENTO DI UNA PROVA D'ESAME COMPLETA ASSEGNAZIONE IN PASSATO

A titolo di esempio, proponiamo lo svolgimento completo della prova d'esame che è stata proposta nell'anno scolastico 2016/2017.

Indirizzo: ITIA - INFORMATICA E TELECOMUNICAZIONI – ARTICOLAZIONE INFORMATICA
Tema di: INFORMATICA - Tipologia C

Il candidato (che potrà eventualmente avvalersi delle conoscenze e competenze maturate attraverso esperienze di alternanza scuola-lavoro, stage o formazione in azienda) svolga la prima parte della prova e due tra i quesiti proposti nella seconda parte.

PRIMA PARTE

Un'azienda start-up vuole costruire una piattaforma web che consenta il car pooling tra viaggiatori sul territorio nazionale, con l'obiettivo di diffondere l'uso di una mobilità flessibile e personalizzata in termini di percorsi e costi. Gli utenti della piattaforma possono essere di due tipi: utenti-autisti (coloro che offrono un passaggio con la propria macchina) e utenti-passeggeri (coloro che usufruiscono del passaggio). Gli autisti devono registrarsi sul sito e inserire i propri dati: generalità, numero e scadenza patente di guida, dati dell'automobile utilizzata, recapito telefonico, e-mail, fotografia. Per ogni viaggio che intendono condividere, gli autisti devono indicare città di partenza, città di destinazione, data e ora di partenza, contributo economico richiesto a ogni passeggero, tempi di percorrenza stimati. È responsabilità dell'autista, mano a mano che accetterà passeggeri per un certo viaggio, dichiarare chiuse le prenotazioni per quel viaggio, utilizzando un'apposita funzione sul portale. L'utente-passeggiere si deve registrare sulla piattaforma, indicando cognome e nome, documento di identità, recapito telefonico ed e-mail. La piattaforma fornisce ai passeggeri la possibilità di indicare città di partenza e di destinazione e data desiderata; presenta quindi un elenco di viaggi (per cui non siano ancora chiuse le prenotazioni), ciascuno con le caratteristiche dell'autista e le modalità del viaggio stesso inserite dall'autista (orario, eventuali soste previste alle stazioni di servizio, possibilità di caricare bagaglio o animali, ...). Il passeggiere sceglie quindi il viaggio desiderato con il corrispondente autista, anche esaminando il voto medio e i giudizi dei feedback assegnati tramite la piattaforma dai precedenti passeggeri all'autista stesso, e si prenota. Le informazioni sul passeggiere vengono inviate per e-mail dalla piattaforma all'autista scelto, il quale può consultare sul portale il voto medio e i giudizi dei feedback ricevuti dal passeggiere da parte di precedenti autisti e decidere se accettarlo o meno. Il passeggiere di conseguenza riceverà una e-mail di accettazione o di rifiuto della prenotazione effettuata, contenente, in caso di accettazione, un promemoria con città di partenza e destinazione, data e orario del viaggio, dati dell'autista e della sua automobile.

A viaggio effettuato, il passeggiere può inserire un feedback sull'autista, espresso sia in forma di voto numerico che di giudizio discorsivo. A sua volta, l'autista può inserire un feedback sul passeggiere, espresso sia in forma di voto numerico che di giudizio discorsivo. Sia i voti medi che i singoli giudizi dei feedback ricevuti da ciascun autista sono disponibili ai passeggeri; analogamente, sia i voti medi che i singoli giudizi dei feedback ricevuti da ciascun passeggiere sono disponibili agli autisti.

Il candidato, fatte le opportune ipotesi aggiuntive, sviluppi:

1. un'analisi della realtà di riferimento, giungendo alla definizione di uno schema concettuale della base di dati che, a suo motivato giudizio, sia idoneo a gestire la realtà presentata;

2. il relativo schema logico;
3. le seguenti interrogazioni espresse in linguaggio SQL:
 - a. data una città di partenza, una di arrivo e una data, elencare gli autisti che propongono un viaggio corrispondente con prenotazioni non ancora chiuse, in ordine crescente di orario, riportando i dati dell'auto e il contributo economico richiesto;
 - b. dato il codice di una prenotazione accettata, estrarre i dati necessari per predisporre l'e-mail di promemoria da inviare all'utente passeggero;
 - c. dato un certo viaggio, consentire all'autista di valutare le caratteristiche dei passeggeri visualizzando l'elenco di coloro che lo hanno prenotato, con il voto medio dei feedback ricevuti da ciascun passeggero, presentando solo i passeggeri che hanno voto medio superiore a un valore indicato dall'autista;
4. il progetto di massima della struttura funzionale dell'applicazione web, realizzando, con appropriati linguaggi a scelta sia lato client che lato server, un segmento significativo dell'applicazione che consente l'interazione con la base di dati.

SECONDA PARTE

- I. In relazione al tema proposto nella prima parte, il candidato integri il modello già realizzato al fine di gestire in automatico il numero di posti disponibili nei vari viaggi, evitando che sia responsabilità dell'autista dichiarare chiuse le prenotazioni sul portale. Nel momento in cui inserisce un viaggio, l'autista dichiara il numero massimo di posti disponibili. Mano a mano che gli autisti accettano le prenotazioni, il sistema visualizzerà solo i viaggi con posti ancora disponibili: a tal fine, una prenotazione non ancora accettata dall'autista non comporta alcun impegno del posto, che resta così ancora disponibile per prenotazioni di altri passeggeri. Per ciascun viaggio, la piattaforma mostrerà il numero dei posti disponibili e il numero delle prenotazioni non ancora accettate. Il candidato sviluppi poi la pagina web, sia lato client che lato server, per fornire ai passeggeri tali informazioni.
- II. In relazione al tema proposto nella prima parte, il candidato immagini di volere documentare al committente l'operatività della piattaforma proposta. A tal fine, imposti una relazione tecnica che presenti le principali caratteristiche dell'applicazione web in termini di organizzazione e funzionalità. In particolare, imposti la struttura di tale relazione, motivando le scelte e scrivendo un esempio significativo dei relativi contenuti.

III. Dato il seguente schema relazionale:

```

    film (id, titolo, durata, anno di produzione, genere);
    attore (id, nome, cognome, data_nascita, fotografia);
    recita (id_film, id_attore, ruolo);
  
```

il candidato:

- determini la modalità di gestione del campo 'fotografia' che prevede la memorizzazione di una immagine dell'attore in un formato grafico (es. JPG);
- formalizzi in linguaggio SQL lo schema fisico corrispondente allo schema relazionale, sapendo che:
 - a. il campo 'genere' ammette solo i seguenti valori: fantasy, giallo, commedia, horror, drammatico, fantascienza, azione;
 - b. per la relazione 'recita', i campi 'id_film' e 'id_attore' referenziano rispettivamente la chiave primaria delle relazioni 'film' e 'attore';
- discuta l'uso degli indici nel modello fisico di una base di dati e suggerisca con motivato giudizio

indici appropriati per questo schema relazionale, definendoli in linguaggio SQL.

- IV. Un'azienda desidera sviluppare un'applicazione web per la prenotazione online di eventi culturali, fruibile sia da computer desktop che da dispositivi mobili come tablet e smartphone. Il candidato esponga i punti critici da affrontare relativamente alle differenti proprietà di visualizzazione delle varie tipologie di dispositivi e alla rispettiva fruizione dei contenuti. Illustri possibili misure risolutive, con esempi relativi all'applicazione in questione.

Durata massima della prova: 6 ore. È consentito l'uso di manuali tecnici e di calcolatrici tascabili non programmabili. È consentito l'uso del dizionario bilingue (italiano-lingua del paese di provenienza) per i candidati di madrelingua non italiana. Non è consentito lasciare l'Istituto prima che siano trascorse 3 ore dalla dettatura del tema.

SVOLGIMENTO DELLA PROVA

PRIMA PARTE

La traccia d'esame ci chiede di gestire un sistema di car pooling (condivisione di auto), che permette di amministrare l'offerta e la richiesta di passaggi in auto tra singoli individui. Negli ultimi anni si è molto diffusa questa modalità di effettuare viaggi, risparmiando sui costi, e sono nate delle vere e proprie applicazioni per gestire questi tipi di spostamenti (per esempio il BlaBlaCar).

Analisi della consegna

Analizzando il problema proposto, ci si rende conto che dobbiamo occuparci della gestione di diversi utenti: gli autisti e i passeggeri.

Gli autisti possono inserire i dati relativi ai viaggi e alle auto impiegate; inoltre confermano le prenotazioni dei passeggeri. I passeggeri possono consultare i dati relativi ai viaggi ed effettuare delle prenotazioni. La prenotazione rimane in sospeso finché non viene confermata dall'autista, che provvederà a decrementare il numero di posti ancora disponibili.

Inoltre a conclusione del viaggio gli autisti possono inserire i giudizi sui passeggeri e i passeggeri possono inserire dei giudizi sull'autista; entrambi gli utenti possono consultare i giudizi che li riguardano e quelli che riguardano gli altri utenti.

Affinché gli utenti possano interagire con il database della compagnia, questo deve essere collocato su un host accessibile dalla rete Internet; deve inoltre essere disponibile una applicazione web che permetta agli utenti di interfacciarsi con la compagnia anche attraverso un dispositivo mobile.

PROGETTAZIONE DI BASI DI DATI - Modellazione concettuale

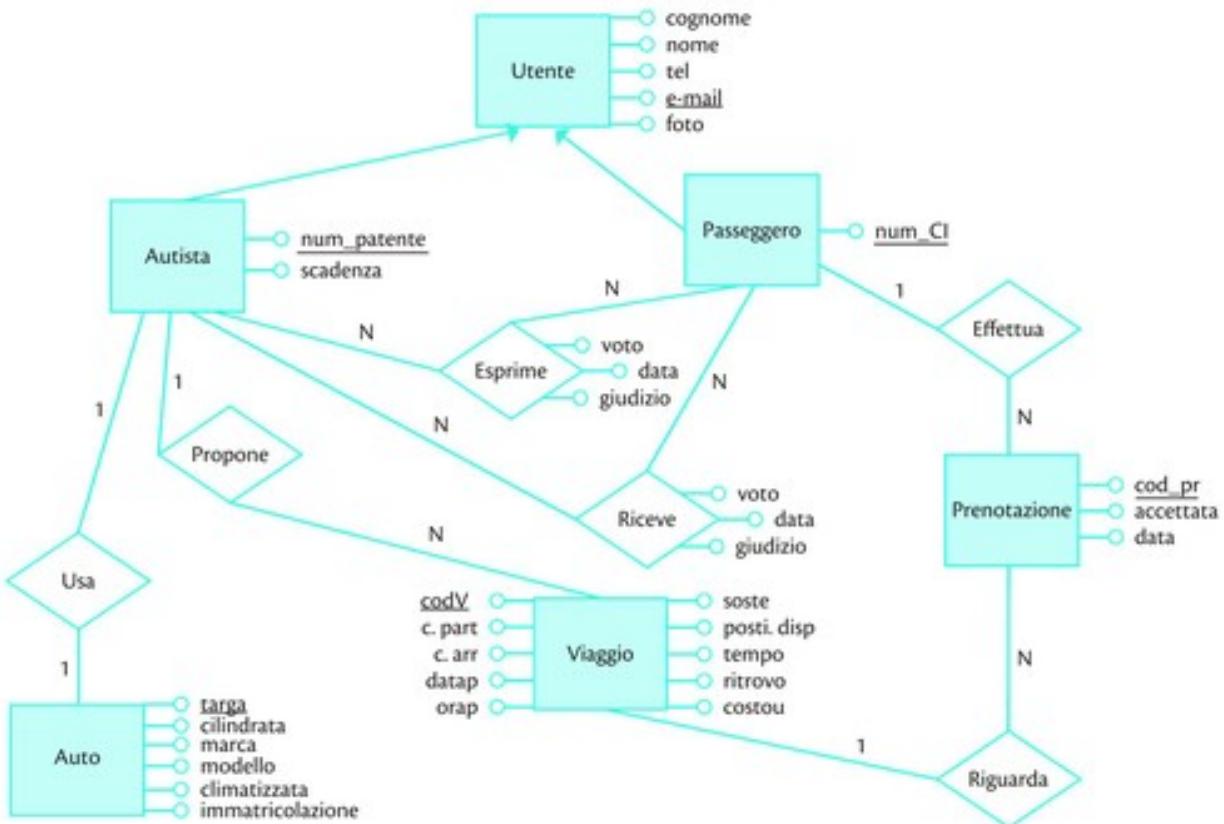
Costruiamo lo schema concettuale che meglio rappresenti i dati di interesse del problema. Il modello E/R comprenderà un'entità padre Utente a cui saranno collegate, tramite una gerarchia di subset, le entità Autista e Passeggero: si tratta di una gerarchia di subset, perché lo stesso utente può essere in alcuni casi autista in altri passeggero. Avremo inoltre le entità Auto, Viaggio e prenotazione.

Per definire gli attributi e le relazioni tra le entità teniamo conto delle richieste del problema.

Per esempio, possiamo inserire due associazioni diverse tra Autista e Passeggero: una indica il giudizio espresso dall'autista sul passeggero (Esprime) e l'altra (Riceve) indica il giudizio espresso da un passeggero sull'autista.

Ipotizziamo che un Autista usi sempre la stessa auto per effettuare i suoi viaggi.

Lo schema E/R proposto rappresenta una possibile soluzione del problema.



Dizionario entità

Nome attributo	Descrizione	Attributi	Identificatore
Utente	Utente passeggero o autista	cognome nome tel e-mail foto datan	e-mail
Passeggero	Utente che chiede un posto per un viaggio	num_CI	num_CI
Autista	Utente che propone un viaggio	num_Patente scadenza	num_Patente
Viaggio	Viaggio proposto	codV c_part c_arr datap orap costou soste posti_disp tempo ritrovo	codV



PREPARAZIONE ALLA SECONDA PROVA SCRITTA DELL'ESAME DI STATO

Auto	Auto utilizzata da un autista per effettuare i viaggi	targa cilindrata marca modello climatizzata immatricolazione	targa
Prenotazione	Una prenotazione richiesta da un passeggero per un viaggio	cod_pr accettata	cod_pr

Dizionario associazioni

Nome associazione	Descrizione	Entità coinvolte	Identificatore
usa	Un autista usa una sola auto per effettuare i viaggi e un'auto è usata da un solo autista	Autista, Auto	
propone	Un autista propone uno o più viaggi, mentre un viaggio è proposto da un unico autista	Autista, Viaggio	
effettua	Un passeggero può effettuare una o più prenotazioni mentre una prenotazione è relativa a un solo passeggero	Passeggero, Prenotazione	
riguarda	Un viaggio può avere più prenotazioni mentre una prenotazione è relativa a un solo viaggio	Viaggio, Prenotazione	
esprime	Indica i giudizi espressi dagli autisti sui passeggeri	Autista, Passeggero	voto giudizio data
riceve	Indica i giudizi espressi dai passeggeri sugli autisti	Autista, Passeggero	voto giudizio data

Dizionario attributi

Utente

Nome attributo	Descrizione	Tipo	Lunghezza	Vincoli
cognome		stringa	30	obbligatorio
nome		stringa	30	obbligatorio
e-mail	Indirizzo e-mail	stringa	30	obbligatorio
tel	Telefono principale	stringa	20	obbligatorio
foto	Link a una foto	stringa	20	obbligatorio
datan	Data di nascita	data	8	obbligatorio

Autista

Nome attributo	Descrizione	Tipo	Lunghezza	Vincoli
num_Patente	Numero della patente di guida	stringa	20	obbligatorio
scadenza	Data di scadenza della patente di guida	data	8	obbligatorio

Prenotazione

Nome attributo	Descrizione	Tipo	Lung.	Vincoli
cod_pr	Codice della prenotazione	Auto increment	20	obbligatorio
accettata	Flag che indica se la prenotazione è stata accettata o meno	boolean		Vero (1)/falso(0)
data	data della prenotazione	data	8	

Passeggero

Nome attributo	Descrizione	Tipo	Lung.	Vincoli
num_CI	Numero della carta di identità	stringa	20	obbligatorio

Viaggio

Nome attributo	Descrizione	Tipo	Lung.	Vincoli
codV		stringa	30	obbligatorio
costou	Contributo richiesto per ogni passeggero	numerico		obbligatorio
datap	Data di partenza	data	8	obbligatorio
orap	Ora di partenza	orario		obbligatorio
ritrovo	Punto di ritrovo	stringa	30	
c_arr	Città di arrivo	stringa	30	obbligatorio
c_part	Città di partenza	stringa	30	obbligatorio
soste	Numero di soste previste	intero		
posti_disp	Numero di posti ancora disponibili	intero		obbligatorio
tempo	Tempo stimato espresso in ore e minuti	stringa	5	

Auto

Nome attributo	Descrizione	Tipo	Lung.	Vincoli
targa	Targa dell'auto	stringa	10	obbligatorio
cilindrata		numerico		
marca		stringa	30	
modello		stringa	20	
climatizzata	Flag che indica l'aria condizionata	boolean		Vero (1)/falso(0)
immatricolazione	Data di immatricolazione	data	8	obbligatorio

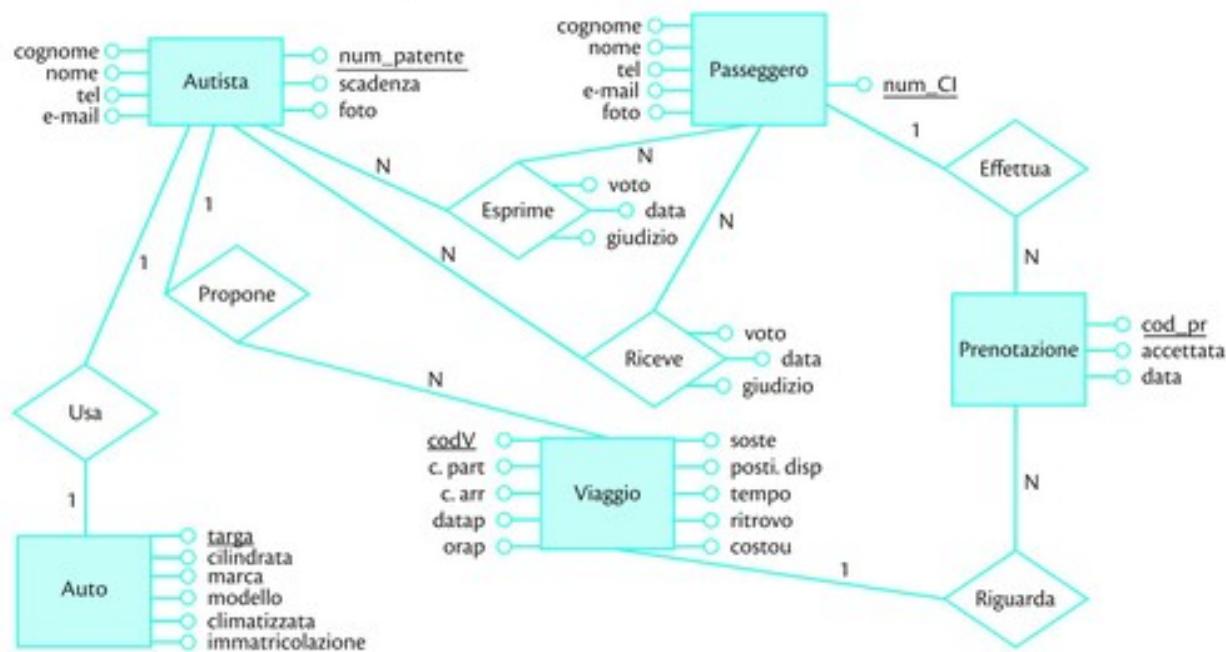
Esprime/riceve

Nome attributo	Descrizione	Tipo	Lung.	Vincoli
voto		numerico		obbligatorio
giudizio		stringa	40	
data	Data di inserimento del giudizio	data	8	obbligatorio

Ristrutturare il modello E/R

Il modello ER ottenuto non può essere tradotto, deve prima essere ristrutturato, perché contiene una gerarchia che deve essere eliminata. Per eliminare la gerarchia di subset sceglieremo di duplicare gli attributi presenti nell'entità padre (Utente) in ciascuna delle entità figlie.

Lo schema ristrutturato sarà il seguente:



A questo punto dal modello E/R ristrutturato passiamo allo schema logico, applicando le regole di derivazione: per ogni entità costruiamo una tabella che avrà gli stessi attributi dell'entità; per realizzare le associazioni presenti nello schema procediamo come illustrato nel seguito.

L'associazione **usa** presente tra Autista e Auto di tipo 1:1 va rappresentata inserendo la chiave primaria di Auto come chiave esterna in Autista (targa_auto).

L'associazione **propone** presente tra Autista e Viaggio di tipo 1:N va rappresentata inserendo la chiave primaria di Autista come chiave esterna in Viaggio (id_autista).

Per realizzare l'associazione di tipo 1:N **effettua** inseriamo la chiave primaria di Passeggero come chiave esterna in Prenotazione (Cl_pass).

Per realizzare l'associazione **riguarda** di tipo 1:N inseriamo la chiave primaria di Viaggio come chiave esterna in Prenotazione (id_Viaggio).

Per realizzare l'associazione di tipo N:N **esprime** costruiamo una nuova tabella che conterrà le chiavi primarie di Passeggero (id_passeggero) e Autista (id_autista) e gli attributi dell'associazione.

Per realizzare l'associazione di tipo N:N **riceve** costruiamo una nuova tabella che conterrà le chiavi primarie di Passeggero (id_passeggero) e Autista (id_autista) e gli attributi dell'associazione.

Modellazione logica

Passeggero (num_CI, cognome, nome, tel, e-mail, foto, datan)
Autista (num_Patente, scadenza, cognome, nome, tel, e-mail, foto, datan, targa_auto)
Auto (targa_cilindrata, marca, modello, climatizzata, immatricolazione)
Viaggio (codV, c_part, c_arr, datap, orap, costou, soste, posti_disp, tempo, ritrovo, id_autista)
Prenotazione (cod_pr, CI_pass, id_Viaggio, accettata)
Esprime (id_autista, id_passeggero, voto, giudizio, data)
Riceve (id_passeggero, id_autista, voto, giudizio, data)

Possiamo notare che il modello relazionale ottenuto è normalizzato, perché tutte le relazioni sono in terza forma normale.

LINGUAGGI PER BASI DI DATI - Scrivere in linguaggio SQL le richieste al database

Sviluppo delle query richieste.

- a. Data una città di partenza, una di arrivo e una data, elencare gli autisti che propongono un viaggio corrispondente con prenotazioni non ancora chiuse, in ordine crescente di orario, riportando i dati dell'auto e il contributo economico richiesto:

```

Select nome, cognome, foto, orap, targa, marca, modello, climatizzata
From auto, autista, viaggio
Where      targa = targa_auto and num_Patente = id_autista and
          posti_disp > 0 and c_part = ['inserire città di partenza']
          and c_arr = ['inserire città di arrivo'] and datap = ['inserire data viaggio']
Order by    orap;
  
```

- b. dato il codice di una prenotazione accettata, estrarre i dati necessari per predisporre l'e-mail di promemoria da inviare all'utente passeggero:

```

Select passeggero.e-mail, viaggio.*, autista.*, auto.*
From auto, autista, viaggio, passeggero, prenotazione
Where      targa = targa_auto and num_Patente = id_autista
          and codv = id_viaggio and num_CI = CI_pass
          and cod_pr = ['inserire codice prenotazione']
          and accettata = 1;
  
```

- c. dato un certo viaggio, consentire all'autista di valutare le caratteristiche dei passeggeri, visualizzando l'elenco di coloro che lo hanno prenotato, con il voto medio dei feedback ricevuti da ciascun passeggero, presentando solo i passeggeri che hanno voto medio superiore a un valore indicato dall'autista:

```

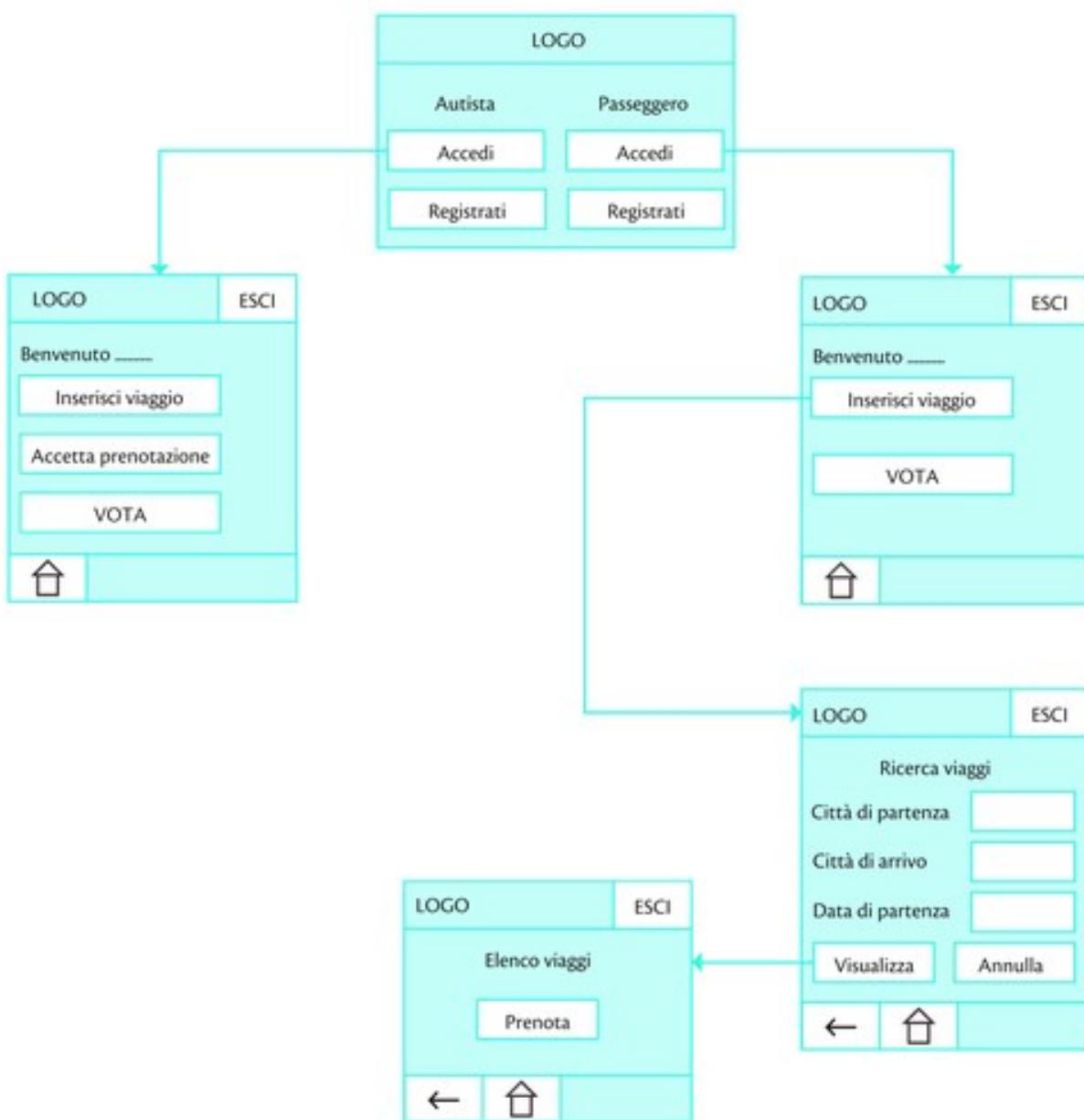
Select id_passeggero, nome, cognome, avg(voto) as voto_medio
From prenotazione, passeggero, esprime
Where      num_CI = CI_pass and num_CI = id_passeggero
          and id_viaggio = ['inserire codice viaggio']
Group by    id_passeggero
Having     avg(voto) > ['inserire voto'];
  
```

**TECNOLOGIE PER IL WEB - Realizzazione del sito web**

Per la risoluzione di questo punto, si può progettare la struttura dell'applicazione web nel suo insieme e implementare la parte relativa alla ricerca di un viaggio da parte di un passeggero e, al successivo inserimento, di una prenotazione.

L'applicazione prevede una serie di pagine che si richiamano tra loro e che hanno tutte la stessa struttura, composta dal logo e da un bottone che permette di tornare alla pagina precedente o alla home page.

Il sito può avere la seguente struttura di massima:



Riportiamo la codifica lato client e lato server dell'inserimento di un viaggio da parte di un autista. È da notare che il codice del viaggio e l'identificativo dell'autista non vengono presi in input, poiché il primo sarà inserito automaticamente e il secondo è già noto. Infatti questa è solo una parte dell'applicazione e si arriva a questa pagina dopo aver fatto l'accesso come autista.

Inserimento viaggio

Città di partenza:

Città di arrivo:

Data di partenza:

Ora di partenza:

Costo:

Soste:

Posti disponibili:

Tempo stimato:

Ritrovo:

CODIFICA PHP

InserimentoViaggio.html

```

<html>
<head></head>
<body>
<p align = "center"><font size = '5'><strong><br>Inserimento viaggio<br>
<form method = 'post' action = 'InsViaggio.php'>
<p align = "left"><font size = '5'>
Città di partenza: <input type = 'text' name = 'cpart' size = '10'><br><br>
Città di arrivo: <input type = 'text' name = 'carr' size = '10'><br><br>
Data di partenza: <input type = 'text' name = 'datap' size = '10'><br><br>
Ora di partenza: <input type = 'text' name = 'orap' size = '10'><br><br>
Costo: <input type = 'text' name = 'costo' size = '10'><br><br>
Soste: <input type = 'text' name = 'soste' size = '10'><br><br>
Posti disponibili: <input type = 'text' name = 'posti' size = '10'><br><br>
Tempo stimato: <input type = 'text' name = 'tempo' size = '10'><br><br>
Ritrovo: <input type = 'text' name = 'ritrovo' size = '10'>
<p align = "center"><font size = '5'>
<input type = 'submit' style="height:50px;font size = '8';" value = 'Inserisci' > &nbsp &nbsp &nbsp &nbsp
&nbsp &nbsp
<input type = 'reset' style="height:50px;" value = 'Annulla'>
</form>
</body>
</html>

```

InsViaggio.php

```
<?php
    $hostname = "localhost";
    $username = "root";
    $password = "";
    $dbname = "carpooling";
    //connessione al server sql
    $conn = mysqli_connect($hostname, $username, $password,$dbname);
    if(!$conn)
        { print "errore nella connessione";
        exit(); }

    //recupera i dati passati dal form html
    $ritrovo = $_POST['ritrovo'];
    $cpart = $_POST['cpart'];
    $carr = $_POST['carr'];
    $datap = $_POST['datap'];
    $costo = $_POST['costo'];
    $orap = $_POST['orap'];
    $soste = $_POST['soste'];
    $posti = $_POST['posti'];
    $tempo = $_POST['tempo'];

    $query = "INSERT INTO viaggio (codV, costou, datap, orap, ritrovo, c_arr, c_part,
        soste, posti_disp, tempo, id_autista)
        VALUES ('v15', '$costo', '$datap', '$orap', '$ritrovo', '$carr', '$cpart',
        '$soste', '$posti', '$tempo', 'bc123df');";
    $risultato= mysqli_real_query($conn,$query);
    if (!$risultato)
        print "errore nell'inserimento";
    else
        print "inserimento avvenuto correttamente";
    mysqli_close($conn);
?>
```

CODIFICA C#.NET**InserimentoViaggio.aspx**

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="InserimentoViaggio.aspx.cs" Inherits="InserimentoViaggio" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
```

```

INSERIMENTO VIAGGIO <br /><br />
<form id="form1" runat="server">
Città di partenza: <asp:TextBox ID="txtCittaP" runat="server">
    </asp:TextBox> <br />
Città di arrivo: <asp:TextBox ID="txtCittaA" runat="server">
    </asp:TextBox><br />
Data di partenza: <asp:TextBox ID="txtDatap" runat="server">
    </asp:TextBox><br />
Ora di partenza: <asp:TextBox ID="txtOraP" runat="server">
    </asp:TextBox><br />
Costo: <asp:TextBox ID="txtCosto" runat="server" Width="67px">
    </asp:TextBox><br />
Soste: <asp:TextBox ID="txtSoste" runat="server" Width="34px">
    </asp:TextBox><br />
Posti disponibili: <asp:TextBox ID="txtPosti" runat="server" Width="29px">
    </asp:TextBox><br />
Tempo stimato: <asp:TextBox ID="txtTempo" runat="server" Width="97px">
    </asp:TextBox><br />
Ritrovo: <asp:TextBox ID="txtRitrovo" runat="server" Width="333px">
    </asp:TextBox><br />
<asp:Button ID="btnInserisci" runat="server" Text="Inserisci"
    OnClick="btnInserisci_Click" Width="100px" />
<asp:Button ID="btnReset" runat="server" Text="Annulla"
    OnClick="btnReset_Click" /> <br />
<asp:Label ID="lblMessaggio" runat="server" Text=""></asp:Label>
</form>
</body>
</html>

```

InserimentoViaggio.aspx.cs (database ACCESS)

```

using System;
using System.Data.OleDb;

public partial class InserimentoViaggio : System.Web.UI.Page
{
    protected void btnInserisci_Click(object sender, EventArgs e)
    {
        OleDbConnection connDb = new OleDbConnection();
        OleDbCommand command = new OleDbCommand();
        connDb.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;" +
            "Data Source=|DataDirectory|/Carpooling.accdb";
        connDb.Open();
        //recupera i dati passati dal form
        String ritrovo = txtRitrovo.Text;
        String c_part = txtCittaP.Text;
        String c_arr = txtCittaA.Text;
        DateTime dataP = Convert.ToDateTime(txtDatap.Text);
        int costo = int.Parse(txtCosto.Text);
    }
}

```

```
DateTime oraP = Convert.ToDateTime(txtOraP.Text);
int soste = int.Parse(txtSoste.Text);
int posti_disp = int.Parse(txtPosti.Text);
String tempo = txtTempo.Text;

string comando = " INSERT INTO viaggio(codV, costou, datap, orap, ritrovo, " +
    " c_arr, c_part, soste, posti_disp, tempo, id_autista)" +
    " VALUES('v15'," + costo + "," + dataP + "#," + oraP + "#," + ritrovo + "," +
    " " + c_arr + "," + c_part + "," + soste + "," + posti_disp +
    ", " + tempo + ", 'bc123df')";
command.Connection = connDb;
command.CommandText = comando;
try
{
    command.ExecuteNonQuery();
    lblMessaggio.Text = "Inserimento avvenuto correttamente ";
}
catch (Exception ex)
{
    lblMessaggio.Text = "Errore durante il caricamento:" + ex.Message;
}
connDb.Close();
}

}
```

InserimentoViaggio.aspx.cs (database SQL Server)

```
using System;
using System.Data.SqlClient;

public partial class InserimentoViaggio : System.Web.UI.Page
{
    protected void btnInserisci_Click(object sender, EventArgs e)
    {
        SqlConnection connDb = new SqlConnection();
        SqlCommand command = new SqlCommand();
        String ritrovo = txtRitrovo.Text;
        String c_part = txtCittaP.Text;
        String c_arr = txtCittaA.Text;
        DateTime dataP = Convert.ToDateTime(txtDatap.Text);
        //Conversione della data nel formato ANSI standard yyyy.mm.dd
        string dataPartenza = dataP.Year + "." + dataP.Month + "." + dataP.Day;
        int costo = int.Parse(txtCosto.Text);
        string oraP = txtOraP.Text;
        int soste = int.Parse(txtSoste.Text);
        int posti_disp = int.Parse(txtPosti.Text);
```

```
String tempo = txtTempo.Text;
connDb.ConnectionString = "Data Source = (LocalDB)\MSSQLLocalDB; " +
    " AttachDbFileName =|DataDirectory|Carpooling.mdf; Initial Catalog = Carpooling.mdf";
connDb.Open();
string comando = " INSERT INTO viaggio(codV, costou, datap, orap, ritrovo, " +
    " c_arr, c_part, soste, posti_disp, tempo, id_autista)" +
    " VALUES('v15'," + costo +
    ", convert(datetime, " + dataPartenza + ", 102) " +
    ", convert(datetime, " + oraP + ", 108)," + ritrovo + "," +
    " " + c_arr + "," + c_part + "," + soste + "," + posti_disp +
    ", " + tempo + ", 'bc123df')";
command.CommandText = comando;
command.Connection = connDb;
try
{
    command.ExecuteNonQuery();
    lblMessaggio.Text = "Inserimento avvenuto correttamente";
}
catch (Exception ex)
{
    lblMessaggio.Text = "Errore durante il caricamento:" + ex.Message;
}
connDb.Close();
}
}
```

SECONDA PARTE

Quesito 1

In relazione al tema proposto nella prima parte, il candidato integri il modello già realizzato al fine di gestire in automatico il numero di posti disponibili nei vari viaggi, evitando che sia responsabilità dell'autista dichiarare chiuse le prenotazioni sul portale. Nel momento in cui inserisce un viaggio, l'autista dichiara il numero massimo di posti disponibili. Mano a mano che gli autisti accettano le prenotazioni, il sistema visualizzerà solo i viaggi con posti ancora disponibili: a tal fine, una prenotazione non ancora accettata dall'autista non comporta alcun impegno del posto, che resta così ancora disponibile per prenotazioni di altri passeggeri. Per ciascun viaggio, la piattaforma mostrerà il numero dei posti disponibili e il numero delle prenotazioni non ancora accettate. Il candidato svilupperà poi la pagina web, sia lato client che lato server, per fornire ai passeggeri tali informazioni.

Nello schema E/R presentato come soluzione nella prima parte della prova si è già tenuto conto dell'inserimento da parte dell'autista del numero di posti disponibili, che potrà essere decrementato ogni volta che viene accettata una prenotazione, tenendo conto che le prenotazioni non possono essere più accettate se il numero di posti è uguale a zero. Nel seguito viene presentata la codifica lato client e lato server della pagina web, che presenta il numero di posti disponibili e il numero di prenotazioni non ancora accettate.

CODIFICA PHP**viaggio.html**

```
<html>
<head></head>
<body>
<p align = "center"><font size = '5'><strong><br>Ricerca viaggi<br><br>
<form method = 'post' action = 'viaggio.php'>
<p align = "center"><font size = '5'>
Città di partenza: <input type = 'text' name = 'cpart' size = '10'><br><br>
Città di arrivo: <input type = 'text' name = 'carr' size = '10'><br><br>
Data di partenza: <input type = 'text' name = 'datap' size = '10'><br><br>
<p align = "center"><font size = '5'>
<input type = 'submit' style="height:50px;font size = '8';" value = 'Visualizza' > &ampnbsp &ampnbsp &ampnbsp
<input type = 'reset' style="height:50px;" value = 'Annulla'>
</form>
</body>
</html>
```

viaggio.php

```
<?php
$hostname = "localhost";
$username = "root";
$password = "";
$dbname = "carpooling";
//connessione al server sql
$conn = mysqli_connect($hostname, $username, $password,$dbname);
if(!$conn)
{
    { print "errore nella connessione";
    exit(); }

//recupera i dati passati dal form html
$cpart = $_POST['cpart'];
$carr = $_POST['carr'];

$datap = $_POST['datap'];
$query = "Select count(*) as prenotati, autista.* ,viaggio.* 
from viaggio,autista,prenotazione
where id_viaggio = codV and id_autista = num_Patente and
c_part = '$cpart' and c_arr = '$carr' and datap='$datap' and accettata = 0
group by codV
order by orap";
$risultato= mysqli_query($conn,$query);
if (!$risultato)
{
    print "errore nel comando";
    exit();
}
```

```

$riga = mysqli_fetch_array($risultato);
if($riga) {
    print "ora partenza &nbsp &nbsp &nbsp";
    print "cognome &nbsp &nbsp &nbsp";
    print "nome &nbsp &nbsp &nbsp &nbsp &nbsp";
    print "costo &nbsp &nbsp &nbsp &nbsp &nbsp &nbsp";
    print "data &nbsp &nbsp &nbsp &nbsp &nbsp &nbsp";
    print "p.disponibili &nbsp &nbsp &nbsp";
    print "p.prenotati &nbsp &nbsp &nbsp";
    print "<br> <br>";
    while($riga) {
        print $riga['orap']."&nbsp &nbsp &nbsp &nbsp &nbsp";
        print $riga['cognome']."&nbsp &nbsp &nbsp &nbsp &nbsp";
        print $riga['nome']."&nbsp &nbsp &nbsp &nbsp &nbsp";
        print $riga['costou']."&nbsp &nbsp &nbsp &nbsp &nbsp";
        print $riga['datap']."&nbsp &nbsp &nbsp &nbsp &nbsp";
        print $riga['posti_disp']."&nbsp &nbsp &nbsp &nbsp &nbsp &nbsp &nbsp";
        print $riga['prenotati']."&nbsp &nbsp &nbsp &nbsp &nbsp &nbsp";
        print "<br>";
        $riga = mysqli_fetch_array($risultato);
    }
}
else
{
    print "Attenzione!!! nessun viaggio presente";
}
mysqli_close($conn);
?>

```

CODIFICA C#.NET

Viaggio.aspx

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Viaggio.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<br />RICERCA VIAGGI<br /><br /><br />
Città partenza<asp:TextBox ID="TxtCittaP" runat="server"></asp:TextBox>
<br /><br />
Città arrivo<asp:TextBox ID="TxtCittaA" runat="server"></asp:TextBox>
<br /><br />

```

```
Data <asp:TextBox ID="TxtData" runat="server" Width="396px"></asp:TextBox>
<br /><br />
<asp:Button ID="BtnVis" runat="server" Text="visualizza " OnClick="BtnVis_Click" />
<br /><br />
<asp:GridView ID="GrdViaggio" runat="server" AutoGenerateColumns="False">
<Columns>
    <asp:BoundField HeaderText="Ora" DataField="orap"/>
    <asp:BoundField HeaderText="cognome " DataField="cognome"/>
    <asp:BoundField HeaderText="Nome " DataField="nome"/>
    <asp:BoundField HeaderText="Costo " DataField="costo"/>
    <asp:BoundField HeaderText="Posti disponibili " DataField="posti_disp"/>
    <asp:BoundField HeaderText="Posto prenotati " DataField="prenotati"/>
</Columns>
</asp:GridView>
<br /><br />
<asp:Label ID="lblMess" runat="server" Text=""></asp:Label>
<br />
</div>
</form>
</body> </html>
```

Viaggio.aspx.cs (database ACCESS)

```
using System;
using System.Data.OleDb;
using System.Data;

public partial class _Viaggio : System.Web.UI.Page
{
    protected void BtnVis_Click(object sender, EventArgs e)
    {
        GrdViaggio.DataSource = null;
        GrdViaggio.DataBind();
        lblMess.Text = "";
        OleDbConnection connDb = new OleDbConnection();
        OleDbCommand command = new OleDbCommand();
        OleDbCommand command1 = new OleDbCommand();
        OleDbDataReader rsViaggio;
        DataTable dt = new DataTable();
        string cittaP = TxtCittaP.Text;
        string cittaA = TxtCittaA.Text;
        DateTime dataPart = Convert.ToDateTime(TxtData.Text);
        connDb.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;" +
            "Data Source=|DataDirectory|/Carpooling.accdb";
        connDb.Open();
        string query =
            "Select orap,cognome,nome,costou,datap,posti_disp,
            (Select count(*) as Prenota " +
```

```

    "from prenotazione,viaggio " +
    "where Prenotazione.id_Viaggio = viaggio.codV " +
    "and V.codV = viaggio.codV " +
    "and accettata = 0 ]" +
    " from viaggio as V, autista " +
    " where V.id_autista = autista.num_Patente " +
    " and c_arr = " + cittaA + " " +
    " and c_part = " + cittaP + " " +
    " and Year(datap) = " + dataPart.Year +
    " and Month(datap) = " + dataPart.Month +
    " and Day(datap) = " + dataPart.Day +
    " order by orap";
command.Connection = connDb;
command.CommandText = query;
rsViaggio = command.ExecuteReader();
if (rsViaggio.HasRows)
{
    dt.Columns.Add("cognome");
    dt.Columns.Add("nome");
    dt.Columns.Add("orap");
    dt.Columns.Add("costo");
    dt.Columns.Add("posti_disp");
    dt.Columns.Add("prenotati");
    while (rsViaggio.Read())
    {
        DataRow row = dt.NewRow();
        row["cognome"] = rsViaggio["cognome"];
        row["nome"] = rsViaggio["nome"];
        DateTime data1 = Convert.ToDateTime(rsViaggio["orap"]);
        row["orap"] = data1.ToString("H:mm");
        row["costo"] = rsViaggio["costou"].ToString();
        row["posti_disp"] = rsViaggio["posti_disp"].ToString();
        row["prenotati"] = rsViaggio[6].ToString();
        dt.Rows.Add(row);
    }
    GrdViaggio.DataSource = dt;
    GrdViaggio.DataBind();
    rsViaggio.Close();
    connDb.Close();
}
else
    lblMess.Text = "viaggio inesistente";
}
}

```

```
using System;
using System.Data.SqlClient;
using System.Data;

public partial class _Viaggio : System.Web.UI.Page
{
    protected void BtnVis_Click(object sender, EventArgs e)
    {
        string oraPart;
        GrdViaggio.DataSource = null;
        GrdViaggio.DataBind();
        lblMess.Text = "";
        SqlConnection connDb = new SqlConnection();
        SqlCommand command = new SqlCommand();
        SqlCommand command1 = new SqlCommand();
        SqlDataReader rsViaggio;
        DataTable dt = new DataTable();
        string cittaP = TxtCittaP.Text;
        string cittaA = TxtCittaA.Text;
        DateTime dataPart = Convert.ToDateTime(TxtData.Text);
        connDb.ConnectionString = "Data Source = (LocalDB)\MSSQLLocalDB; " +
        " AttachDbFileName =|DataDirectory|Carpooling.mdf; Initial Catalog = Carpooling.mdf";
        connDb.Open();
        string query =
            "Select orap,cognome, nome, costou, datap, posti_disp, " +
            " (Select count(*) from prenotazione,viaggio" +
            " where Prenotazione.id_Viaggio = viaggio.codV " +
            " and V.codV = viaggio.codV " +
            " and accettata = 0 )" +
            " from viaggio as V, autista " +
            " where V.id_autista = autista.num_Patente " +
            " and c_arr = " + cittaA + " " +
            " and c_part = " + cittaP + " " +
            " and Year(datap) = " + dataPart.Year +
            " and Month(datap) = " + dataPart.Month +
            " and Day(datap) = " + dataPart.Day +
            " order by orap";
        command.Connection = connDb;
        command.CommandText = query;
        rsViaggio = command.ExecuteReader();
        if (rsViaggio.HasRows)
        {
            dt.Columns.Add("cognome");
            dt.Columns.Add("nome");
            dt.Columns.Add("orap");
            dt.Columns.Add("costo");
            dt.Columns.Add("soste");
            dt.Columns.Add("posti_disp");
            dt.Columns.Add("prenotati");
```

```

while (rsViaggio.Read())
{
    DataRow row = dt.NewRow();
    row["cognome"] = rsViaggio["cognome"];
    row["nome"] = rsViaggio["nome"];
    DateTime dataP = Convert.ToDateTime(rsViaggio["orap"].ToString());
    if( dataP.Minute == 0)
        oraPart = dataP.Hour + ":" + dataP.Minute + "0";
    else
        oraPart = dataP.Hour + ":" + dataP.Minute;
    row["orap"] = oraPart;
    row["costo"] = rsViaggio["costou"].ToString();
    row["posti_disp"] = rsViaggio["posti_disp"].ToString();
    row["prenotati"] = rsViaggio[6].ToString();
    dt.Rows.Add(row);
}
GrdViaggio.DataSource = dt;
GrdViaggio.DataBind();
rsViaggio.Close();
connDb.Close();
}
else
    lblMess.Text = "viaggio inesistente";
}

}

```

Viaggio.aspx.cs (database SQL Server)

Esecuzione

Ricerca viaggi

Città di partenza:

Città di arrivo:

Data di partenza:



Quesito II

In relazione al tema proposto nella prima parte, il candidato immagini di volere documentare al committente l'operatività della piattaforma proposta. A tal fine, imposti una relazione tecnica che presenti le principali caratteristiche dell'applicazione web in termini di organizzazione e funzionalità. In particolare, imposti la struttura di tale relazione, motivando le scelte e scrivendo un esempio significativo dei relativi contenuti.

La documentazione di un progetto è una parte molto importante e va realizzata durante tutta la sua durata.

Esistono vari tipi di documenti che accompagnano il progetto. I principali sono:

- requisiti utente e specifiche iniziali contenenti le richieste del cliente;
- architettura del sistema e specifiche funzionali contenenti il progetto della struttura del prodotto e le funzionalità che verranno rese disponibili. È il documento che accompagna il contratto tra cliente e fornitore;
- specifiche di dettaglio e di implementazione contenenti il progetto del sistema e, successivamente, come è stato implementato;
- *reference manual* con i dettagli di implementazione del sistema, da utilizzare nella fase di manutenzione successiva;
- manuale utente per mostrare all'utente finale le modalità di uso del sistema.

Molto spesso, per comodità di rapporti con il cliente, viene realizzata una relazione riassuntiva contenente in sintesi alcune delle informazioni presenti in dettaglio nei vari documenti: è quello che viene richiesto dalla traccia. In particolare viene chiesto che la relazione contenga informazioni sulla struttura architettonica e su quelle funzionali.

L'indice della relazione potrebbe essere il seguente.

1. Requisiti del cliente: verranno descritte le richieste sia dal punto di vista architettonico che funzionale. In pratica è il testo della prova.
2. Architettura del sistema: viene descritta la struttura del sistema, l'organizzazione dei client e dei server.
3. Struttura del sito:
 - a. funzionalità previste: vengono elencate le funzionalità che il sistema rende disponibile con una breve descrizione delle stesse;
 - b. interfaccia utente: viene mostrato il layout dell'interfaccia e le modalità di interazione dell'utente.
4. Architettura del database: viene documentata la struttura logica del database.

Nello svolgimento della nostra prova queste parti sono quasi tutte già presenti. In particolare il punto 1 corrisponde al testo della prova, il punto 4 è il progetto logico del database, mentre il punto b del terzo punto è la descrizione del sito web.

Sviluppiamo quindi l'elenco delle funzionalità previste dal sistema.

Le funzionalità sono divise in base al tipo di utente:

- autista
 -
- passeggero.

AUTISTA

1. Registrazione: registrazione come nuovo autista.
2. Accesso: accesso tramite login e password come autista.
3. Viaggio:
 - a. inserimento viaggio: creazione di un nuovo viaggio e inserimento dei dati relativi;
 - b. conferma prenotazione: permette di visualizzare le prenotazioni effettuate per un proprio viaggio, ed eventualmente di confermarle.
4. Valutazioni:
 - a. valuta passeggero: inserimento della valutazione di un passeggero;
 - b. valutazione passeggero: permette di vedere il punteggio di un passeggero;
 - c. valutazione autista: permette di vedere il punteggio ottenuto come autista.

PASSEGGERO

1. Registrazione: registrazione come nuovo passeggero.
2. Accesso: accesso tramite login e password come passeggero.
3. Viaggio:
 - a. cerca viaggio;
 - b. prenotazione: permette di prenotare un viaggio.
4. Valutazioni:
 - a. valuta autista: inserimento della valutazione di un autista;
 - b. valutazione passeggero: permette di vedere il punteggio ottenuto come passeggero;
 - c. valutazione autista: permette di vedere il punteggio di un autista.

Quesito III

Dato il seguente schema relazionale:

*film (id, titolo, durata, anno di produzione, genere);
attore (id, nome, cognome, data_nascita, fotografia);
recita (id_film, id_attore, ruolo);*

il candidato:

- determini la modalità di gestione del campo 'fotografia' che prevede la memorizzazione di una immagine dell'attore in un formato grafico (es. JPG);
- formalizzi in linguaggio SQL lo schema fisico corrispondente allo schema relazionale, sapendo che:
 - a. il campo 'genere' ammette solo i seguenti valori: fantasy, giallo, commedia, horror, drammatico, fantascienza, azione;
 - b. per la relazione 'recita', i campi 'id_film' e 'id_attore' referenziano rispettivamente la chiave primaria delle relazioni 'film' e 'attore';
- discuta l'uso degli indici nel modello fisico di una base di dati e suggerisca con motivo giudizio indici appropriati per questo schema relazionale, definendoli in linguaggio SQL.

Anche se i diversi DBMS danno la possibilità di inserire le immagini all'interno della base di dati usando opportuni tipi di dati (Oggetto OLE in Access, image per SQL Server, BLOB per mySQL), questa è una tecnica sconsigliata per la pesantezza dei dati all'interno del database. È preferibile invece inserire nel database non l'immagine vera e propria, ma solo il percorso in cui l'immagine è stata salvata: in questo modo il campo 'fotografia' sarà di tipo testo.

```
CREATE TABLE film [  
    id      VARCHAR(5) PRIMARY KEY,  
    titolo  VARCHAR(50),  
    durata  FLOAT,  
    anno    INT,  
    genere  VARCHAR(15) CHECK (genere IN  
        ('giallo','commedia', 'horror', 'drammatico', 'fantascienza', 'azione '))];  
  
CREATE TABLE attore[  
    id      INT PRIMARY KEY,  
    nome   VARCHAR(20),  
    cognome  VARCHAR(20),  
    data_nascita DATE,  
    fotografia  VARCHAR(50)  
  
CREATE TABLE recita [  
    id      INT PRIMARY KEY,  
    id_film  VARCHAR(5),  
    id_attore  VARCHAR(5),  
    ruolo   VARCHAR(30),  
    FOREIGN KEY id_film REFERENCES film(id),  
    FOREIGN KEY id_attore REFERENCES attore(id)]
```

Un indice è una struttura dati realizzata per migliorare i tempi di ricerca delle query. Così come si consulta l'indice di un libro per trovare più velocemente le pagine che trattano di un certo argomento, allo stesso modo, se viene creato un indice su un campo di un database, la ricerca risulterà più veloce perché si evita la scansione dell'intera tabella.

Per rendere più veloce la ricerca dei film di un determinato genere si può definire un indice proprio sul campo genere:

```
CREATE INDEX indGenere ON film ('genere')
```

L'istruzione `DROP INDEX indGenere` elimina l'indice `indGenere`.

Quesito IV

Un'azienda desidera sviluppare un'applicazione web per la prenotazione online di eventi culturali, fruibile sia da computer desktop che da dispositivi mobili, come tablet e smartphone. Il candidato esponga i punti critici da affrontare relativamente alle differenti proprietà di visualizzazione delle varie tipologie di dispositivi e alla rispettiva fruizione dei contenuti. Illustri possibili misure risolutive, con esempi relativi all'applicazione in questione.

Le problematiche che emergono quando si vuole creare un'applicazione fruibile sia da computer desktop sia da tablet e dispositivi mobili sono di due tipi: fruizione dei contenuti e visualizzazione. Per quanto riguarda la fruizione dei contenuti, cioè la necessità che un contenuto sia facilmente accessibile sia da un dispositivo touchscreen sia da un dispositivo provvisto di mouse, è necessario agire a livello di progettazione. Possiamo notare, per esempio, che utilizzando un dispositivo touchscreen il menu a tendina può essere difficoltoso da usare, per cui è preferibile sostituirlo con dei radio button.

A livello di visualizzazione è necessario tener conto della differenza di dimensione degli schermi su cui l'applicazione dovrà essere utilizzata: è quindi necessario creare applicazioni responsive, ovvero che si adattino automaticamente alla dimensione del dispositivo su cui vengono eseguite.

Per realizzare una pagina web responsiva occorre personalizzare il layout in base alla dimensione dello schermo utilizzato, fissando e definendo i cosiddetti breakpoint: cioè definendo a livello di CSS i valori limite, espressi in pixel, entro i quali si verifica una modifica del layout della pagina.

In particolare in CSS3 possiamo utilizzare l'attributo @media, come nell'esempio:

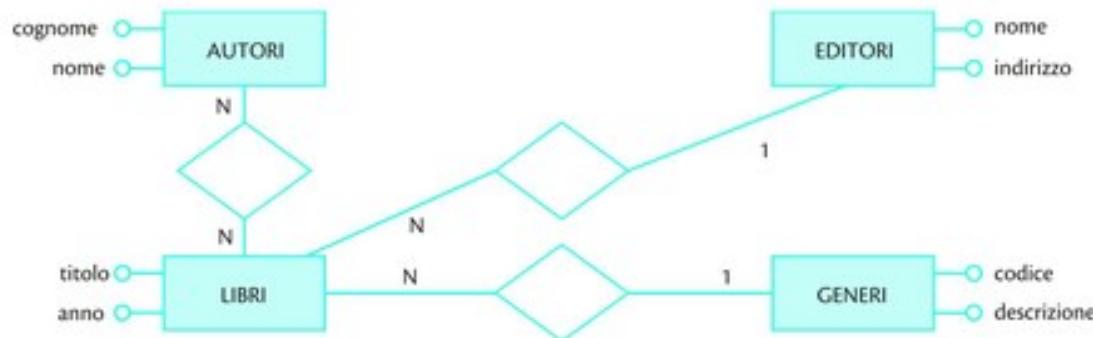
```
@media only screen and (max-device-width: 767px) {  
    .pagina {  
        width: 350px;  
        margin-top: 10px;  
        ...  
    }  
}
```

In tal caso solo se il device è screen e la dimensione massima è di 767px(breakpoint), alla classe 'pagina' si applica lo stile width:350px e margin-top:10px. Lo stesso procedimento può essere utilizzato per modificare qualunque altra proprietà. Questi tipi di funzionalità sono dette media queries.

ESERCIZI FINALI

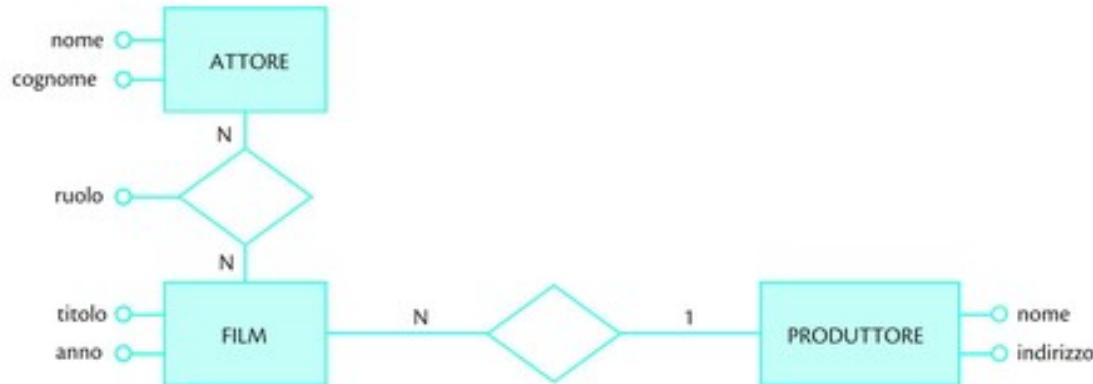
Progettazione di basi di dati

1. Lo schema concettuale di un database per gestire una libreria è descritto dal seguente modello entità-relazioni:



- Completa lo schema E/R con la descrizione delle entità e delle associazioni.
- Trasformalo in uno schema relazionale, indicando i passi eseguiti.

2. Lo schema concettuale di un database per gestire i film è descritto dal seguente modello entità-relazioni:



- Completa lo schema E/R con la descrizione delle entità e delle associazioni.
 - Trasformalo in uno schema relazionale, indicando i passi eseguiti.
3. Una ditta è organizzata in reparti e ciascun dipendente può lavorare in un unico reparto al quale è assegnato un unico caporeparto. I dipendenti della ditta possono frequentare corsi di aggiornamento.
Il database realizzato deve rispondere alle seguenti interrogazioni:
- elenco dei dati anagrafici dei dipendenti di un reparto;
 - elenco dei dipendenti che sono stati assunti nell'intervallo compreso tra due date;
 - elenco dei reparti con il relativo caporeparto;

- d. numero di dipendenti per ciascun reparto;
e. elenco di tutti i dipendenti che hanno seguito un corso di aggiornamento.
4. Progetta un piccolo database per gestire le informazioni relative ai registi, ai film e agli attori di una casa produttrice seguendo questo schema di lavoro:
■ analisi delle richieste per individuare entità, attributi e relazioni (definizione dello SCHEMA);
■ breve descrizione delle entità e degli attributi e definizione delle chiavi;
■ definizione dello schema logico.
a. stampa tutte le informazioni per il film intitolato "Cattivissimo me" (titolo, durata, genere, anno);
b. stampa la durata e il genere del film "Una notte da leoni";
c. stampa il titolo di tutti i film dell'attore "Orlando Bloom";
d. stampa il titolo, il genere e l'anno per i film interpretati da "Orlando Bloom";
e. stampa il nome e l'anno di nascita per gli attori che hanno titolo di studio "Orlando Bloom Perito Informatico";
f. stampa il nome dei registi con cui ha lavorato l'attrice "Emma Watson";
g. stampa tutti i titoli dei film diretti dal regista "Sofia Coppola";
h. inserisci un nuovo film con tutte le informazioni legate.
5. Si devono raccogliere e organizzare le informazioni relative ai risultati nelle gare sportive di un campionato internazionale che si svolge in diverse località del mondo in date diverse nell'anno (per esempio la Coppa del mondo di sci o il Campionato mondiale di F1).
In ogni gara i concorrenti possono guadagnare punti, che alla fine del campionato determinano il vincitore. Individua le entità, gli attributi e le associazioni, motivando le scelte effettuate, disegna il modello E/R. Definisci poi lo schema logico utilizzando le regole di derivazione.
6. Costruisci un modello di dati per la gestione delle informazioni riguardanti gli immobili e i loro proprietari; un immobile può essere intestato anche a più proprietari (con relative percentuali di possesso) e naturalmente una persona può possedere più immobili.
Gli immobili sono di diverse tipologie: abitazioni, uffici, negozi.
La gestione deve ottenere:
■ le informazioni anagrafiche dei proprietari di un determinato immobile con la relativa percentuale di possesso;
■ l'elenco degli immobili con i relativi proprietari che si trovano in una determinata via;
■ le caratteristiche degli immobili di una certa tipologia con metratura, piano, numero locali.
7. Le informazioni sulle opere d'arte di molti artisti di epoche diverse, opere conservate nei musei delle nazioni di tutto il mondo, devono essere catalogate e registrate in un archivio. In una stessa città possono esserci più musei. Le opere possono essere di tipo diverso: tele, sculture ecc. Uno stesso artista può avere opere in tanti musei, così come in un museo, in genere, ci sono opere di artisti diversi. Per sapere l'epoca di riferimento dell'artista si potrebbe registrare la sua data di nascita e la

data di morte (quest'ultima è vuota se l'artista è vivente). Progetta lo schema E/R, la struttura logica e le principali richieste che si possono fare.

8. Un amministratore di stabili vuole memorizzare in un database le informazioni relative ai diversi condomini che amministra. In ciascun condominio ci sono tanti alloggi e per suddividere correttamente le spese condominiali è necessario memorizzare il piano in cui si trova l'alloggio, la superficie in mq e il numero di componenti della famiglia che vi abita. È inoltre necessario conoscere il nome dei consiglieri del condominio (al massimo 2).

Ogni alloggio può essere di proprietà di più persone, ciascuna con la propria percentuale di possesso; comunque l'amministratore deve sapere a chi inviare le comunicazioni in caso di multiproprietà.

L'amministratore, per la manutenzione dei condomini, si avvale di personale specializzato (elettricisti, muratori, idraulici e così via) che esegue dei lavori nei condomini e per ciascun lavoro deve conoscere la data di inizio, la data di conclusione e il numero di ore impiegate, per poter pagare l'operaio secondo una paga oraria definita per ogni categoria (elettricista, muratore e così via).

Per ogni lavoro è inserita una breve descrizione (per esempio "Riparazione caldaia"). In ogni momento l'amministratore deve avere la possibilità di conoscere i dati anagrafici degli operai.

9. Crea un database con informazioni legate alla criminalità di una città. Considera in particolare:

- i criminali e la loro divisione in bande;
- i tipi di crimine e i crimini che ciascun criminale (o banda) ha commesso;
- le vittime dei crimini;
- le zone in cui si sono consumati.

Prevedi le seguenti richieste:

- a. elenco dei criminali appartenenti a una data banda;
- b. numero di crimini commessi da un dato criminale;
- c. età media dei criminali che hanno commesso un dato tipo di crimine;
- d. inserire un nuovo criminale;
- e. modificare la zona di azione di una data banda;
- f. elenco dei crimini di cui sono state vittime persone piemontesi;
- g. elenco dei criminali che agiscono in una data zona;
- h. tipo di crimini commessi da criminale con età inferiore alla media.

10. Si voglia automatizzare la gestione delle lauree triennali dell'università di Torino. L'università è suddivisa in varie facoltà e ogni facoltà ha più corsi di laurea. Uno studente può conseguire la laurea triennale solo se ha raggiunto un numero di crediti maggiore o uguale a 180 e se ha presentato la tesi o la relazione sul tirocinio svolto. Ogni esame ha un numero di crediti prefissato (un credito circa per ogni dieci ore di lezione). Nel corso di un anno accademico uno studente può totalizzare al massimo 80 crediti. Al momento dell'iscrizione a ogni studente viene assegnato un numero di matricola al quale si fa sempre riferimento. Un docente può tenere uno o più insegnamenti (ciclo di lezioni con esame finale), mentre un insegnamento può essere tenuto da più docenti diversi.

- Prevedi le seguenti richieste:
- a. stampa dei dati relativi agli studenti che hanno preso 30 all'esame di "Sistemi";
 - b. stampa delle informazioni su tutti gli esami sostenuti da uno studente;
 - c. stampa dell'elenco degli studenti che possono laurearsi;
 - d. stampa dell'elenco dei corsi di laurea di una data facoltà;
 - e. elenco degli studenti di un determinato corso di laurea;
 - f. stampa dei nomi e dei crediti relativi ai corsi tenuti da un dato docente;
 - g. stampa del numero di crediti totalizzato da un dato studente nel corso di un determinato anno accademico;
 - h. stampa della situazione attuale di un dato studente (numero di crediti totalizzati, tesi o tirocinio conclusi).
11. Si voglia automatizzare la gestione delle Olimpiadi invernali considerando i dati relativi agli atleti, alle gare, agli impianti e al villaggio olimpico. Le richieste da soddisfare sono le seguenti:
- a. stampa di nome e cognome degli atleti che partecipano alle gare di "salto" che si svolgono a "Pragelato";
 - b. stampa di numero di palazzina, numero di camera, data di arrivo e di partenza di un dato atleta;
 - c. stampa di giorno e ora di svolgimento delle gare a cui partecipa un dato atleta;
 - d. visualizzazione di nome, cognome e gara degli atleti impegnati nelle gare che si svolgono in una data località;
 - e. stampa di nome, cognome e nazionalità degli atleti che partecipano alle gare di una data specialità;
 - f. visualizzazione dei dati relativi agli alloggi (numero camera, settore, giorni di permanenza) degli atleti tedeschi.
12. Il Salone del Gusto è organizzato in stand, identificati da un numero; di ogni stand interessa la zona in cui è situato, la superficie, il nome dell'azienda espositrice. In uno stand sono esposti vari prodotti, identificati da un codice unico, e di ogni prodotto si conoscono la tipologia, le caratteristiche ecc. Di ogni prodotto si possono fare degli assaggi. A ogni visitatore viene assegnato un codice, e di ogni visitatore viene registrato il nome, l'età e quali assaggi sono stati fatti. Progetta lo schema E/R, lo schema logico e le principali richieste che si possono fare.
13. Si vogliono gestire le informazioni sugli acquisti dei clienti abituali di una catena di supermercati. Dei clienti interessano il codice fiscale, i dati anagrafici, (comprensivi di sesso, anno di nascita e indirizzo, completo di città, Provincia e Regione) nonché il reddito. Considera le spese effettuate da ciascun cliente, la data, il totale della spesa, la modalità del pagamento (carta, bancomat, contanti) e, per ogni prodotto, la quantità, il prezzo pagato e lo sconto.
Di ogni prodotto interessano il codice, la descrizione, la categoria, il costo unitario e il prezzo di vendita. I prodotti possono essere in promozione, con riduzione temporanea del prezzo, a partire da una certa data e per un numero prefissato di giorni. Progetta lo schema E/R, lo schema logico e le principali richieste che si possono fare.

Il linguaggio SQL

14. Date le seguenti tabelle:

PRODOTTO (Codice, Descrizione, Prezzo, Quantità, Categoria, Fornitore)

codice : Codice del prodotto

descrizione : Descrizione del prodotto

prezzo : Prezzo del prodotto

quantità : Quantità di prodotto

categoria : Categoria merceologica del prodotto

fornitore : Codice del fornitore

FORNITORE (Cod_F, Nome, Indirizzo, Città)

cod_f : Codice del fornitore

nome : Nome del fornitore

indirizzo : Indirizzo del fornitore

città : Città del fornitore

scrivi i comandi SQL per soddisfare le seguenti richieste:

- a. elencare tutti i prodotti in ordine di codice;
- b. stampare nome e indirizzo di tutti i fornitori;
- c. stampare l'indirizzo del fornitore Stam;
- d. stampare il nome dei prodotti con prezzo superiore a 500;
- e. stampare il nome e l'indirizzo del fornitore dei prodotti la cui quantità è inferiore a 100;
- f. stampare la descrizione dei prodotti il cui codice inizia per C;
- g. stampare il numero dei prodotti raggruppati per fornitore;
- h. trovare e visualizzare il prodotto con prezzo massimo;
- i. elencare i prodotti forniti dalla ditta Pincopal;
- j. incrementare di 100 unità il prodotto con codice H74;
- k. aumentare del 5% il prezzo dei prodotti della categoria F;
- l. inserire un nuovo fornitore;
- m. stampare codice e descrizione dei prodotti il cui prezzo è compreso tra 1000 e 5000;
- n. stampare l'elenco dei prodotti forniti da fornitori di Milano;
- o. stampare per ogni categoria la descrizione, il prezzo e la quantità dei prodotti ordinati per prezzo;
- p. caricare in una nuova tabella la descrizione dei prodotti e il nome del fornitore per prodotti di categoria B o C.

15. Date le seguenti tabelle:

FIUME (NomeF, Portata, Navigabile)

nomef : Nome del fiume

portata : Portata del fiume

navigabile : Vale "vero" se il fiume è navigabile, "falso" in caso contrario

PROVINCIA (NomeP, Nab, NomeR)
nomep : Nome della Provincia
nab : Numero di abitanti della Provincia
nomer : Nome della Regione di appartenenza

ATTRaversa (NomeF, NomeP, Km)
nomef : Nome del fiume
nomep : Nome della Provincia attraversata dal fiume
km : Numero di km del fiume nella Provincia

scrivi i comandi SQL per soddisfare le seguenti richieste:

- a. visualizzare il nome di tutte le Province;
- b. visualizzare il nome di tutte le Regioni;
- c. visualizzare il nome di tutti i fiumi;
- d. visualizzare il nome di tutti i fiumi suddivisi in navigabili e non;
- e. visualizzare le Province che si trovano in una determinata Regione;
- f. visualizzare il numero di Province suddiviso per Regione;
- g. visualizzare il numero di abitanti per una determinata Regione;
- h. visualizzare la media degli abitanti per ogni Regione;
- i. visualizzare il numero di abitanti per ogni Regione;
- j. visualizzare il nome dei fiumi che attraversano una determinata Provincia;
- k. visualizzare il nome delle Province attraversate da un determinato fiume;
- l. visualizzare il nome delle Regioni attraversate da un determinato fiume;
- m. visualizzare la lunghezza totale di un determinato fiume;
- n. visualizzare il nome della Provincia che ha il maggior numero di abitanti;
- o. visualizzare il nome del fiume che attraversa più Regioni;
- p. visualizzare il nome delle Province non attraversate da fiumi;
- q. visualizzare il nome dei fiumi che sono lunghi più di 80 km;
- r. caricare in una nuova tabella il nome dei fiumi che attraversano più di quattro Province;
- s. modificare la lunghezza di un determinato fiume;
- t. aumentare del 10% il numero di abitanti delle Province che si trovano in una determinata Regione.

Pagine web

16. Crea una pagina web che presenti il nome della tua scuola (molto grande e in grassetto) e l'indirizzo sulla riga successiva con carattere più piccolo. Il titolo della pagina dev'essere "La mia scuola".
17. Nella pagina creata per l'esercizio precedente inserisci un'immagine e l'elenco delle attività che si svolgono nella scuola.
18. Crea una pagina che permetta di collegarsi ai siti ufficiali di quattro diversi cantanti.

Basi di dati

19. Un Istituto Tecnico Commerciale vuole gestire in un database i dati degli studenti. Per ogni studente si vogliono memorizzare in una tabella i dati della tabella studenti:
- crea una tabella Access con i dati elencati cercando di dettagliare il più possibile le caratteristiche dei campi (tipo e proprietà);
 - carica un certo numero di record;
 - crea una maschera per l'inserimento dei dati; aggiungi sulla maschera l'età dello studente calcolata come differenza tra la data del giorno e la data di nascita diviso 365.
- (Nota: la differenza tra date si può fare come operazione normale e fornisce il numero di giorni che intercorrono tra le due date, dividendo per 365 si ottiene il numero di anni);
- crea il report dell'elenco studenti, raggruppati per classe e sezione, in ordine alfabetico di cognome e nome. Per ogni studente riporta solo i seguenti dati: identificatore, cognome, nome, data di nascita, classe, sezione e religione;
 - crea le seguenti query:
 - tutti gli studenti delle classi quinte indirizzo Informatica;
 - tutti gli studenti maschi di una classe e sezione richieste in input in ordine di cognome e nome.

Per entrambe le query estrai, per ogni studente, solo i seguenti dati: identificatore, cognome, nome, data di nascita, classe, sezione e religione.

Tabella studenti

campo	descrizione	regola di convalida
Identificatore studente	numero che identifica in modo univoco ciascuno studente. È la chiave primaria	generato automaticamente (numero progressivo)
Cognome	cognome dello studente	richiesto
Nome	nome dello studente	richiesto
Indirizzo	indirizzo dello studente	
CAP	codice di avviamento postale	
Città	città dove abita	lo studente
Sesso	indica se lo studente è maschio o femmina	richiesto, può assumere solo i valori 'M' o 'F'
Telefono	numero di telefono dello studente	
Data di nascita	data di nascita dello studente	
Classe	classe frequentata dallo studente (solo il numero)	richiesto, può assumere solo valori compresi tra 1 e 5
Sezione	sezione di al più 2 caratteri (per es. A, B5)	
Indirizzo corso di studi	indirizzo del corso di studi (per es. Mercurio, Igea ecc.)	richiesto
Religione	indica se lo studente fa religione sì o no	
Note	spazio dove inserire informazioni particolari sullo studente	

20. Data la tabella fatture così strutturata:

Tabella fatture		
campo	descrizione	regola di convalida
Numero fattura	numero della fattura (chiave primaria)	generato automaticamente (numero progressivo)
Data fattura	data di emissione della fattura	richiesto
Imponibile	importo imponibile	richiesto
Importo IVA	importo IVA	richiesto
Ragione sociale del cliente	cliente a cui la fattura è intestata	richiesto

- a. crea la struttura della tabella fatture;
- b. carica un certo numero di dati;
- c. crea le seguenti query:
 - tutte le fatture emesse successivamente a una data richiesta e con importo inferiore a 1000. Per ogni fattura riportare id_fattura, data fattura, importo totale fattura;
 - tutte le fatture del cliente "Rossi Mario". Per ogni fattura riportare id_fattura, data fattura, importo totale fattura;
 - tutte le fatture di un cliente di cui viene richiesta in input la ragione sociale (inserirne anche solo una parte). Per ogni fattura riportare id_fattura, data fattura, importo totale fattura;
- d. crea i seguenti report:
 - tutte le fatture emesse nel mese di febbraio. Per ogni fattura riportare id_fattura, data fattura, importo totale fattura, ragione sociale del cliente;
 - tutte le fatture emesse in un mese richiesto in input (in numero). Per ogni fattura riportare id_fattura, data fattura, importo totale fattura, ragione sociale del cliente;
 - tutte le fatture emesse da non più di 30 giorni. Per ogni fattura riportare id_fattura, data fattura, importo totale fattura, ragione sociale del cliente.

PHP e ASP.NET

I seguenti esercizi possono essere fatti utilizzando pagine ASP o pagine PHP.

21. Tramite un form HTML viene fornito il nome di una persona e il sesso (M o F); definisci una pagina dinamica che visualizzi il nome della persona preceduto da "Egregio signore" o "Gentilissima Signora" a seconda che si tratti di un uomo o di una donna.
22. Scrivi una pagina dinamica che riceva da un form HTML un numero compreso tra uno e dodici e visualizzi il nome del mese corrispondente in italiano e in inglese.
23. Scrivi una pagina dinamica che determini se il numero fornito da un form HTML è primo o meno. Nel secondo caso visualizza anche i divisori del numero.

24. Tramite un form HTML si acquisisce la data di nascita e il nome di uno studente. Scrivi una pagina dinamica che visualizzi un messaggio di auguri di buon compleanno se lo studente compie gli anni nel giorno corrente.
25. Scrivi una pagina dinamica che visualizzi la tavola pitagorica in forma tabellare.
26. Una palestra vuole progettare un sito web al fine di divulgare le proprie tariffe. Il cliente può sottoscrivere un abbonamento mensile (65 euro), trimestrale (180 euro) o annuale (450 euro) al quale è applicato uno sconto del 10% se si tratta di un abbonamento per studenti (età inferiore a 24 anni) e di un ulteriore sconto del 15% se si intende frequentare la palestra solo al mattino.
27. La tabella ABBONATI di un database contiene le informazioni sugli abbonati di una rivista (numero abbonamento, cognome, nome, indirizzo, numero telefono, data di scadenza). Visualizza la data di scadenza per l'abbonamento di cui si fornisce il numero tramite un form HTML.
28. Facendo riferimento al database dell'esercizio precedente, visualizza l'elenco degli abbonati per i quali è scaduto l'abbonamento.
29. Gestisci il conto corrente bancario tramite un sito web. Il database è costituito dalle tabelle CONTO_CORRENTE che contiene le informazioni riguardanti il conto corrente (numero conto, saldo, nominativo correntista) e dalla tabella MOVIMENTI contenente le operazioni effettuate (numero conto, tipo operazione – Prelievo o Versamento – , importo dell'operazione). Aggiornare la tabella CONTO_CORRENTE dopo aver prelevato da un form HTML il codice del conto corrente, l'importo movimentato e il tipo di operazione.
30. Scrivi una pagina HTML contenente un form che consenta all'utente di inserire nome, password attuale e nuova password e che richiami una pagina che permetta di cambiare la password nella tabella utenti; nel caso l'utente non sia presente nella tabella fai visualizzare un messaggio di errore e torna alla pagina iniziale.

Appendici

1. XAMPP e MySQL

**2. Visual Studio
e SQL Server**

**3. Creare
pagine ASPX**

4. I comandi SQL

1. XAMPP E MYSQL

Per scrivere software in PHP e creare database in MYSQL è possibile utilizzare l'ambiente XAMPP.

XAMPP è un software gratuito e multiplattforma utilizzato principalmente per la creazione di Web Server. Esso integra infatti Apache, MYSQL, PHP, Perl e tanti altri programmi.

Per il momento ci sono quattro distribuzioni XAMPP:

- | | |
|---|--|
| <ul style="list-style-type: none"> ■ Linux; ■ Mac OS X; | <ul style="list-style-type: none"> ■ Windows; ■ Solaris. |
|---|--|

■ Installazione dell'ambiente XAMPP

Per l'ultima release di XAMPP utilizzare il link <http://www.apachefriends.org>, scaricare la versione interessata, quindi effettuare l'installazione. Al momento dell'installazione è possibile selezionare solo i servizi necessari per usare meno spazio su disco, in particolare è essenziale installare Apache, PHP, MYSQL (MariaDB). È inoltre molto utile installare anche **phpMyAdmin**. Una volta effettuata l'installazione, per accedere al pannello di controllo bisogna mandare in esecuzione l'applicazione **xampp-control** presente nella cartella di installazione (default C:\xampp).

Si presenterà la videata di fig. 1.

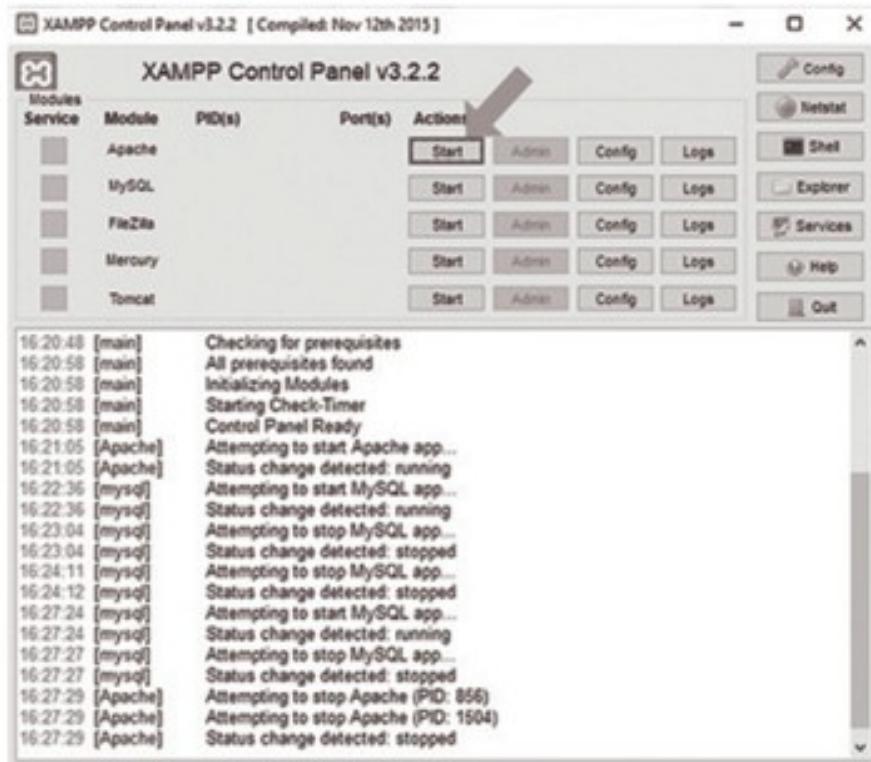
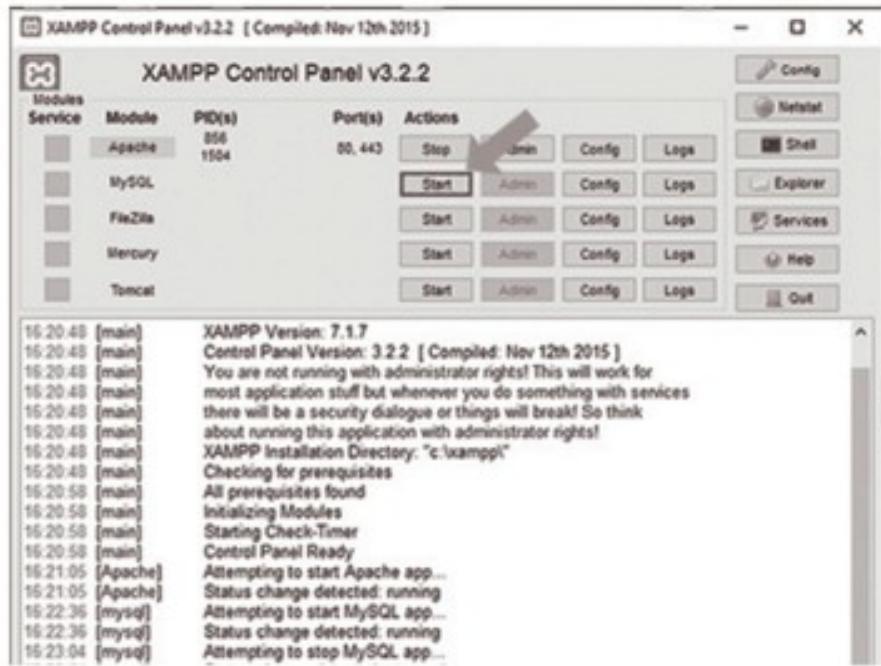


fig. 1 Pannello di controllo XAMPP

fig. 2 Attivazione moduli

Premendo i pulsanti Start accanto ad Apache e MYSQL saranno attivati i moduli relativi (fig. 2).

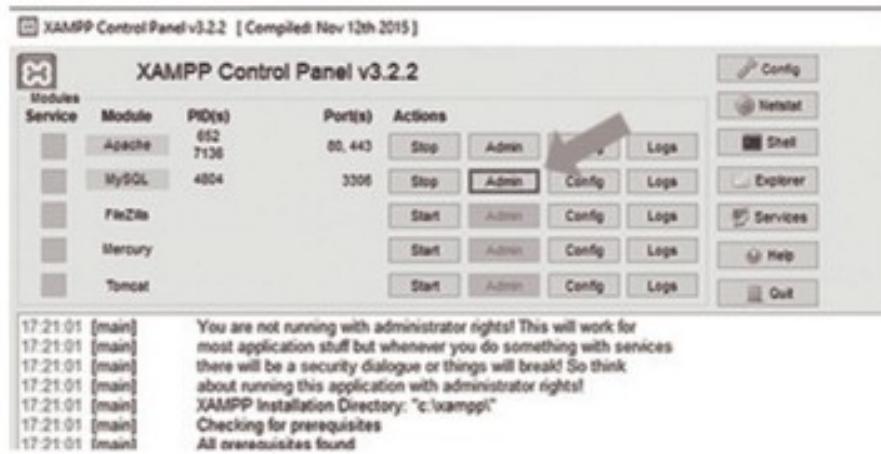


A questo punto XAMPP è attivo. Per utilizzarlo è necessario aprire un browser e digitare il link http://localhost/nome_file, dove nome_file è il nome del file che si vuole eseguire con l'estensione (.php o .html).

Tutti i documenti www e i file PHP che si vogliono creare vanno inseriti nella cartella \xampp\htdocs. Per esempio il file test.html memorizzato nella cartella \xampp\htdocs può essere sfogliato in <http://localhost/test.html> (se il server Apache è in esecuzione).

Si possono anche creare sottocartelle. Per esempio, se si crea la cartella \xampp\htdocs\new e vi si copia il file test.html è necessario inserire l'URL <http://localhost/new/test.html> per vederlo nel browser. Dal pannello di controllo di XAMPP è possibile accedere al phpMyAdmin cliccando sul tasto corrispondente (fig. 3).

fig. 3 Attivazione del phpMyAdmin



Il pannello phpMyAdmin è mostrato nella videata in **fig. 4**.

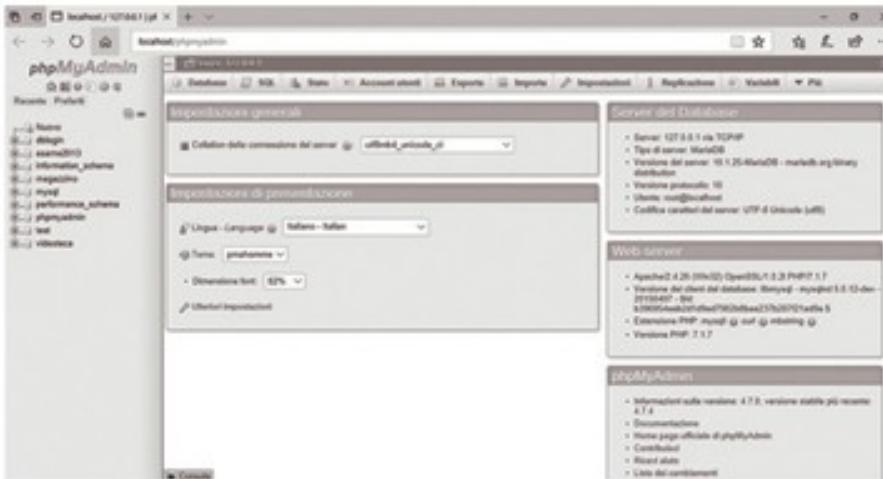


fig. 4 Attivazione del phpMyAdmin

■ Creare un database

Per creare un database è necessario aprire il pannello Database, scrivere il nome del nuovo database e fare Clic sul tasto **Crea** (**fig. 5**).

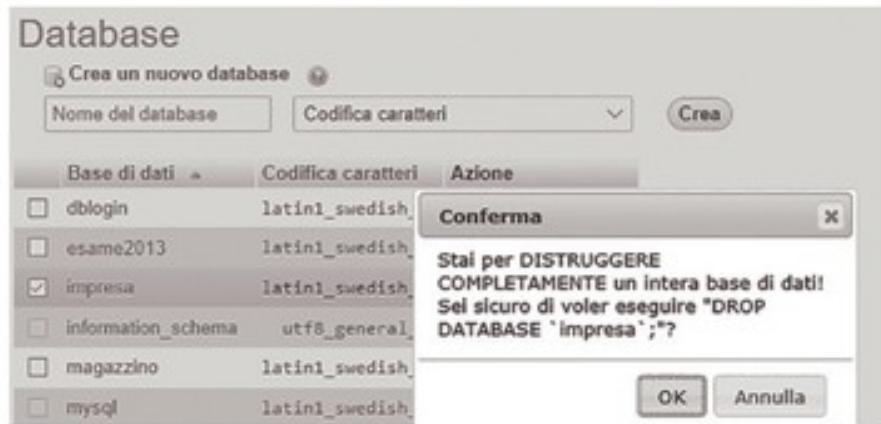
Base di dati	Codifica caratteri	Azione
dblogin	latin1_swedish_ci	Controlla i privilegi
esame2013	latin1_swedish_ci	Controlla i privilegi
information_schema	utf8_general_ci	Controlla i privilegi
magazzino	latin1_swedish_ci	Controlla i privilegi
mysql	latin1_swedish_ci	Controlla i privilegi
performance_schema	utf8_general_ci	Controlla i privilegi
phpmyadmin	utf8_bin	Controlla i privilegi
test	latin1_swedish_ci	Controlla i privilegi
videoteca	latin1_swedish_ci	Controlla i privilegi
Totale: 9	latin1_swedish_ci	Elimina

Per eliminare un database è necessario selezionare il database da eliminare nell'elenco di quelli presenti e premere il pulsante **Elimina**.

fig. 5 Creare un database

fig. 6 Conferma eliminazione

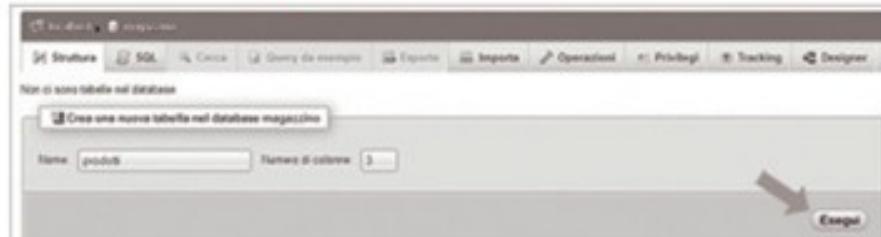
Dando conferma, il database viene eliminato definitivamente (**fig. 6**):



■ Creare una tabella

Una volta creato il database è possibile creare le tabelle che lo compongono.

Per creare una tabella è necessario fare doppio Clic sul nome del database: compare una videata dove è possibile inserire il nome della tabella e il numero di colonne (**fig. 7**):

fig. 7 Creazione tabella

Premendo il pulsante Esegui si apre la finestra mostrata in **fig. 8** in cui è possibile definire la struttura della tabella: nomi dei campi, tipo, chiavi, ecc.

fig. 8 Struttura tabella

Al fondo della videata (fig. 9) è possibile dare conferma delle operazioni effettuate.



fig. 9 Salvataggio struttura tabella

Scegliendo il tasto *Salva* si crea la struttura della tabella e compare la videata di fig. 10.

The screenshot shows the phpMyAdmin interface. On the left, the sidebar displays the database "magazzino" and the table "prodotti". A modal window titled "Crea una nuova tabella nel database magazzino" is open in the center. It has fields for "Nome" (Name) and "Numero di colonne" (Number of columns), both currently empty. An "Esegui" (Execute) button is at the bottom right of the modal. In the background, the main interface shows the table structure for "prodotti" with columns "codice", "descrizione", and "quantità".

fig. 10 Database con tabella

Inserire dati in una tabella

Per inserire dei dati in una tabella occorre selezionare il pulsante *Inserisci* presente nel menu di fig. 10. Comparirà la maschera mostrata in fig. 11, in cui è possibile inserire i dati.

fig. 11 Inserimento dati

The screenshot shows the "Inserisci" (Insert) form for the "prodotti" table. The top navigation bar includes "localhost", "magazzino", and "prodotti". The toolbar buttons are: Mostra, Struttura, SQL, Cerca, Inserisci (highlighted in blue), Esporta, Importa, and Opera. The main area has tabs for "Campo", "Tipo", "Funzione", "Null", and "Valore". Data is entered for three rows:

- Row 1: codice (int(11)) value 1020
- Row 2: descrizione (varchar(50)) value quaderno a quadretti
- Row 3: quantità (int(11)) value 100

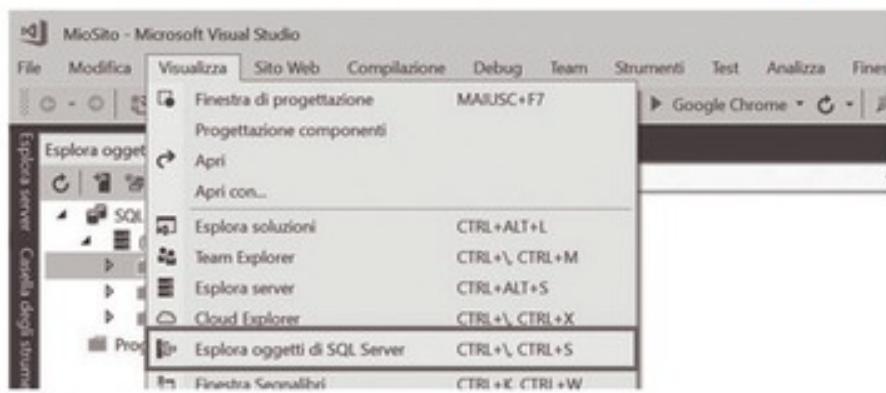
At the bottom, there is an "Esegui" (Execute) button and a checked "Ignora" (Ignore) checkbox. Below the main form, there is a second, empty copy of the insert form.

2. VISUAL STUDIO E SQL SERVER

LAVORARE CON SQL SERVER

Un modo semplice per lavorare con SQL Server è quello di avviare Visual Studio e selezionare *Visualizza > Esplora oggetti di SQL Server*: (fig. 12)

fig. 12 SQL Server



Creare un database

Per creare un nuovo database fare un clic con il tasto destro del mouse sulla voce *Database* e selezionare la voce *Aggiungi nuovo database* (fig. 13)

fig. 13 Creazione un database

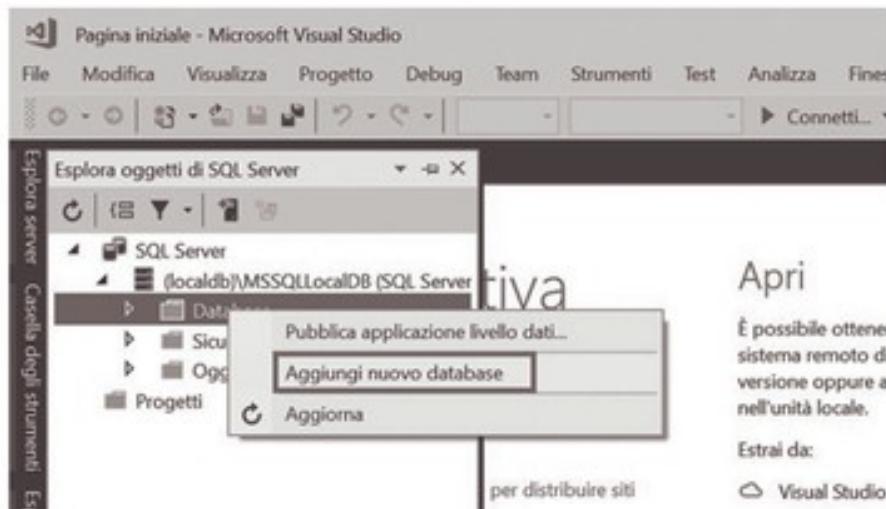
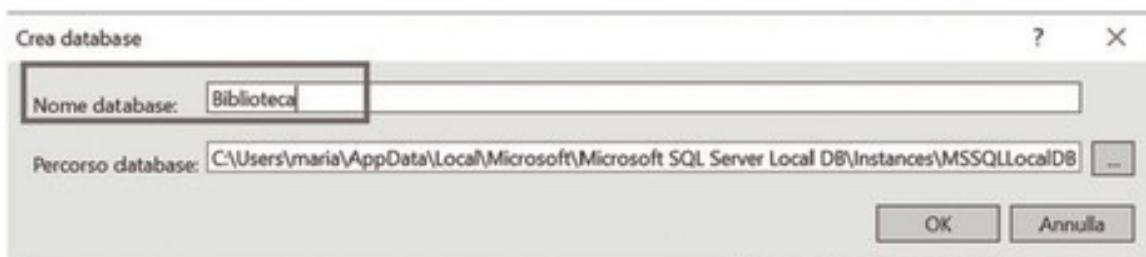


fig. 14 Inserimento nome database

Nella finestra che appare inserire il nome del database (fig. 14):



Per cancellare un database è necessario fare un clic con il tasto destro sul nome del database e selezionare la voce *Elimina* nella finestra che si apre (fig. 15):

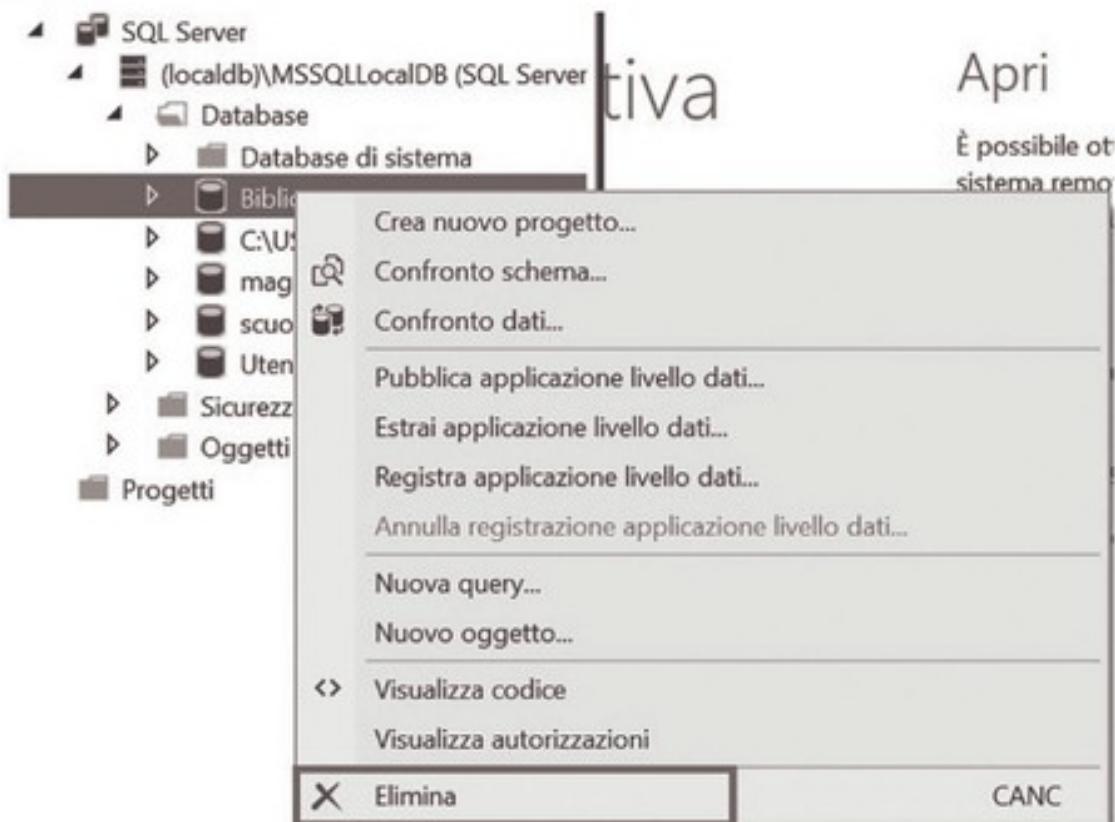


fig. 15 Eliminazione database

Dando la conferma il database viene eliminato definitivamente (fig. 16):



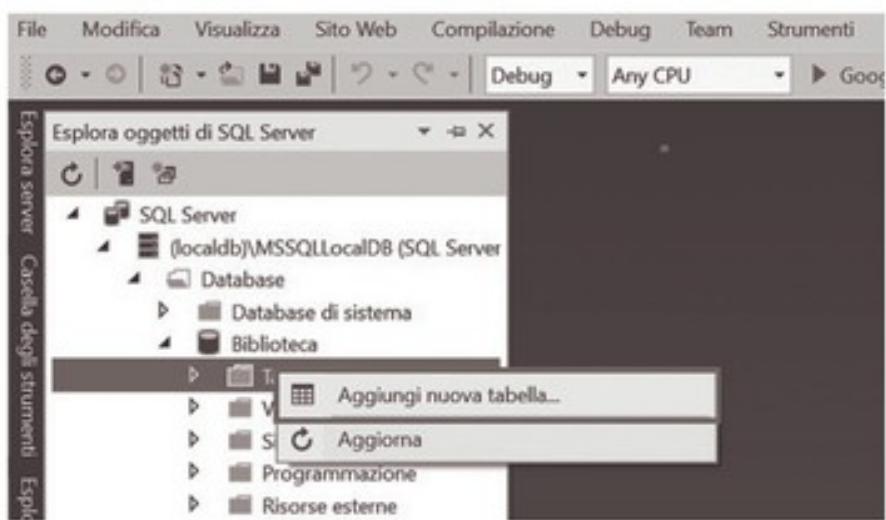
fig. 16 Conferma eliminazione database

Creare una tabella

Una volta creato il database è possibile creare le tabelle che lo compongono. Per creare una tabella è necessario aprire il database con un clic sul triangolino che appare accanto al nome del database stesso.

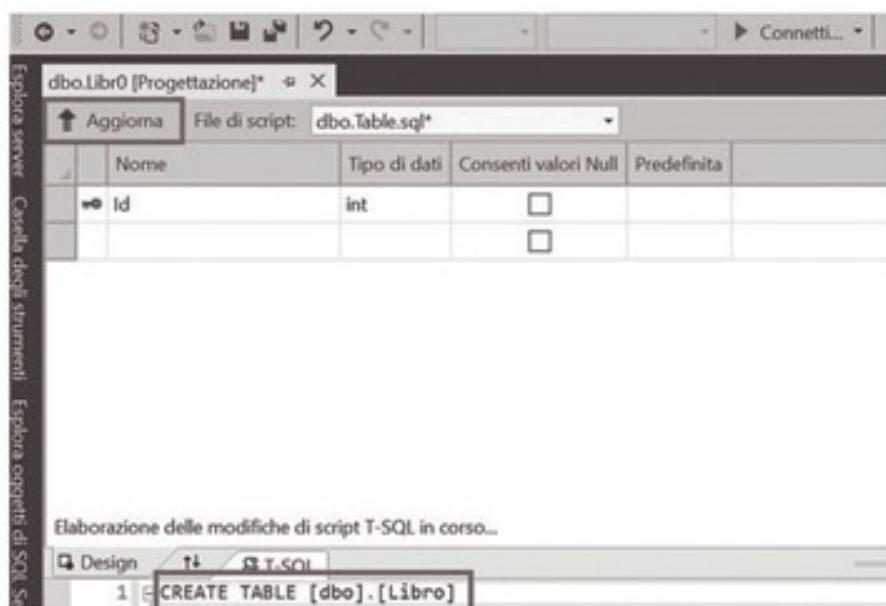
2. VISUAL STUDIO E SQL SERVER

fig. 17 Creazione tabella



La videata che si apre (fig. 17) è divisa in due sezioni: nella parte superiore è possibile definire la struttura della tabella: nomi dei campi, tipo, chiavi ecc., in quella inferiore è presentato il relativo codice SQL. È possibile dare il nome alla tabella scrivendolo direttamente nel comando SQL (fig. 18)

fig. 18 Struttura tabella



Cliccando poi su Aggiorna il nome della tabella comparirà fra quelle presenti nel database. Una volta creata la tabella è possibile ritornare alla videata della sua definizione con un doppio clic sul nome della tabella stessa.

Inserire/visualizzare/eliminare dati in una tabella

Per inserire e per visualizzare le righe di una tabella basta fare un clic col tasto destro del mouse sul nome della tabella interessata: si apre una finestra nella quale si sceglie la voce *Visualizza dati* (fig. 19)

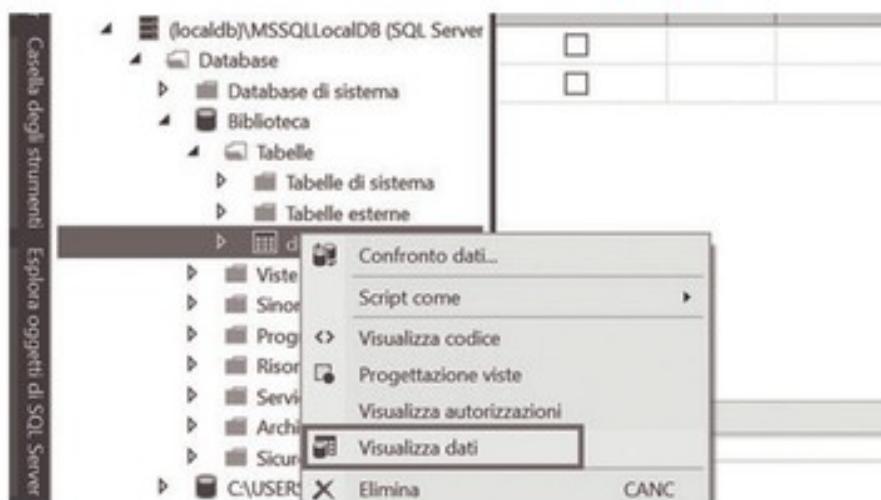


fig. 19 Inserimento/visualizzazione dati

Nella videata che compare si possono inserire nuove righe e visualizzare quelle che sono già presenti (fig. 20).

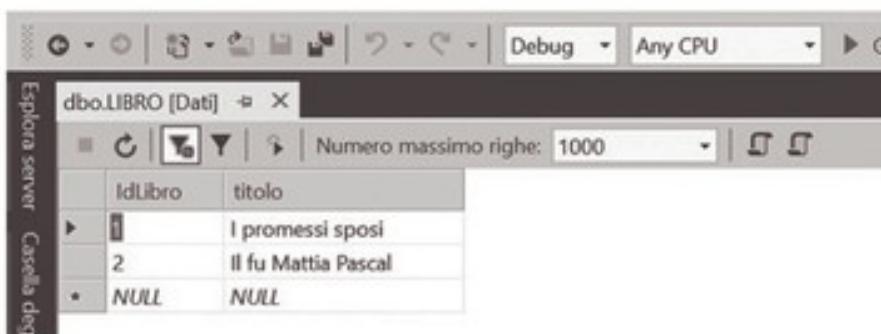


fig. 20 Inserire/visualizzare dati

Per cancellare una riga bisogna posizionarsi su di essa e fare clic col tasto destro del mouse per aprire la finestra di fig. 21 nella quale, scegliendo la voce *Elimina*, si cancella il record. Per eliminare una tabella nella fig. 19 si sceglie la voce *Elimina*.

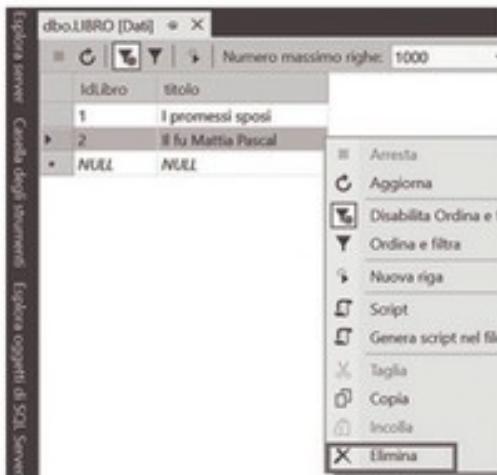


fig. 21 Eliminare una riga

2. VISUAL STUDIO E SQL SERVER

fig. 22 Scrittura query

Realizzare query

È possibile scrivere delle query in linguaggio SQL sul database selezionato [fig. 22](#) e [fig. 23](#):

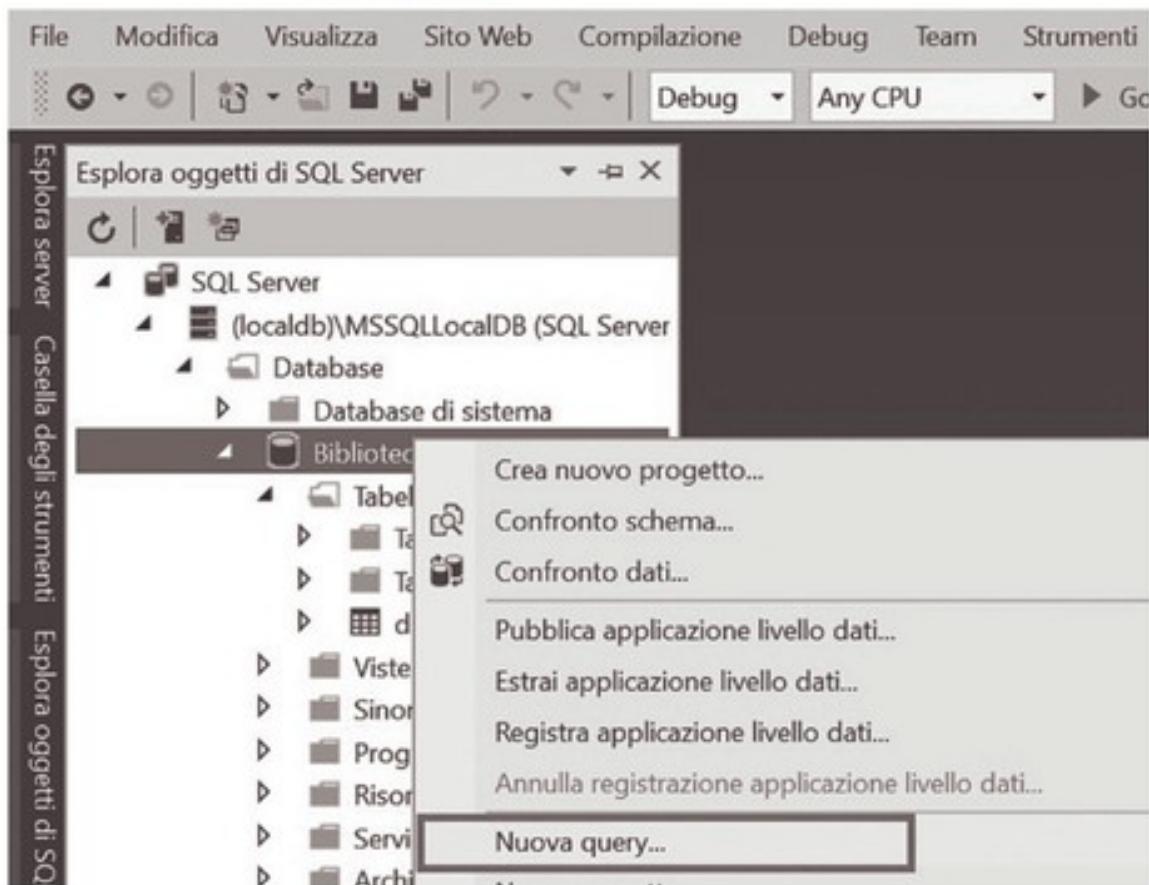
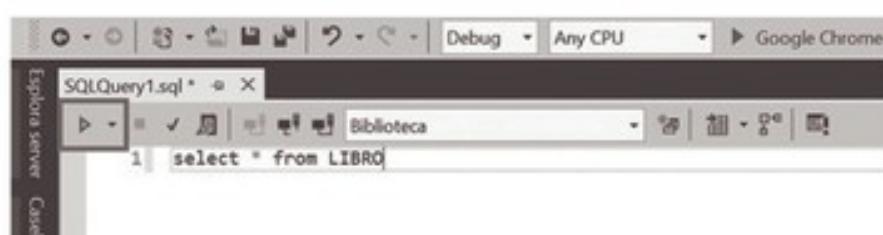


fig. 23 Query SQL



Per eseguire la query fare clic sul triangolino verde. Eseguendo la query si avrà il risultato mostrato in [fig. 24](#):

fig. 24 Risultato della query SQL

	IdLibro	titolo
1	1	I promessi sposi
2	2	Il fu Mattia Pascal

3. CREARE PAGINE ASPX

ASP.NET (*Active Server Page*) è il linguaggio che, sfruttando la tecnologia .NET, permette di scrivere delle pagine web dinamiche eseguite sul server e di creare dei veri e propri applicativi web (*Web Application*) che garantiscono alte performance, sicurezza e versatilità.

Il codice ASP.NET viene compilato (a differenza del codice ASP che viene interpretato) per cui la visualizzazione da parte del browser può risultare lenta la prima volta che viene attivata, ma risulteranno più veloci le visualizzazioni successive.

Come è noto, una pagina dinamica viene eseguita dal server, il quale costruisce una pagina HTML, con all'interno dati eventualmente prelevati da un database, e la invia al client che la visualizza attraverso un browser (fig. 25).

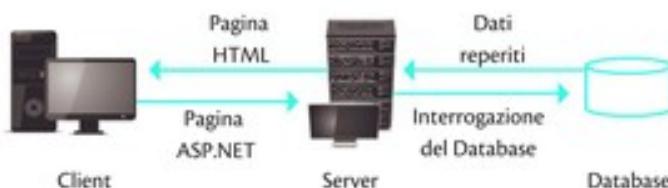


fig. 25
Visualizzazione
di una pagina
ASP.NET

Per lavorare con ASP.NET occorre che il computer diventi un Web Server, e per questo ASP.NET mette a disposizione dell'utente due modalità tramite cui il computer può diventare un **server web locale**: il software IIS (*Internet Information Service*) della Microsoft e il server web integrato nell'ambiente di sviluppo **Visual Web Developer**.

Nel seguito utilizzeremo Microsoft Visual Studio 2015, scaricabile dal sito della Microsoft e il linguaggio VB.NET.

Lo **sviluppatore ASP.NET** può fare affidamento su un nuovo approccio per la realizzazione di pagine web. Nella programmazione vecchio stile, infatti, lo sviluppatore doveva per prima cosa cimentarsi con i dettagli di HTML prima di poter mettere a punto una pagina web dinamica. In quella situazione, per modificare la pagina aggiungendo nuovi contenuti, era necessario lavorare sui tag HTML.

ASP.NET risolve il problema utilizzando un modello più avanzato che provvede a generare il codice HTML in modo automatico. L'ambiente di sviluppo comprende un'interfaccia visuale all'interno della quale è possibile inserire i controlli che interessano, associando eventualmente il codice da eseguire lato server. Il file HTML viene creato automaticamente a seconda degli oggetti che l'utente inserisce nell'interfaccia; tale file avrà estensione **.aspx**.

Nella cartella del progetto sarà presente un altro file con estensione

.aspx.vb (se il linguaggio utilizzato è il VB.NET) con le istruzioni da eseguire.

Quando una pagina di un sito web è richiamata da se stessa è necessario capire se si tratta della prima richiesta o se la pagina ha già effettuato un richiamo (*post*) a se stessa. Da questo dipendono le operazioni successive. Prima di ASP.NET, nei form venivano usati dei campi nascosti (*hidden*) per poter riconoscere se si trattava della prima chiamata della pagina o di chiamate successive.

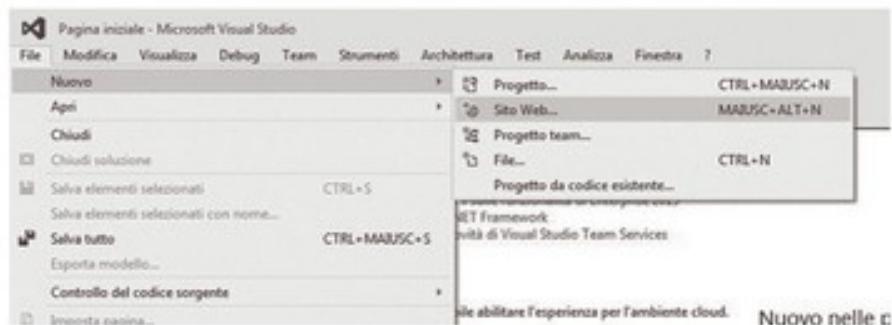
Questa tecnica è chiamata *post back*. Il post back è implementato in ASP.NET in modo invisibile al programmatore. La proprietà *Page.IsPostBack* mostra appunto se una pagina è stata caricata per la prima volta oppure in seguito a un postback.

■ Creare un sito web

Vediamo ora i passi da seguire per creare un sito web.

Innanzitutto è necessario avviare Visual Studio 2015 e cliccare su *File* → *Nuovo* → *Sito web* come illustrato in fig. 26:

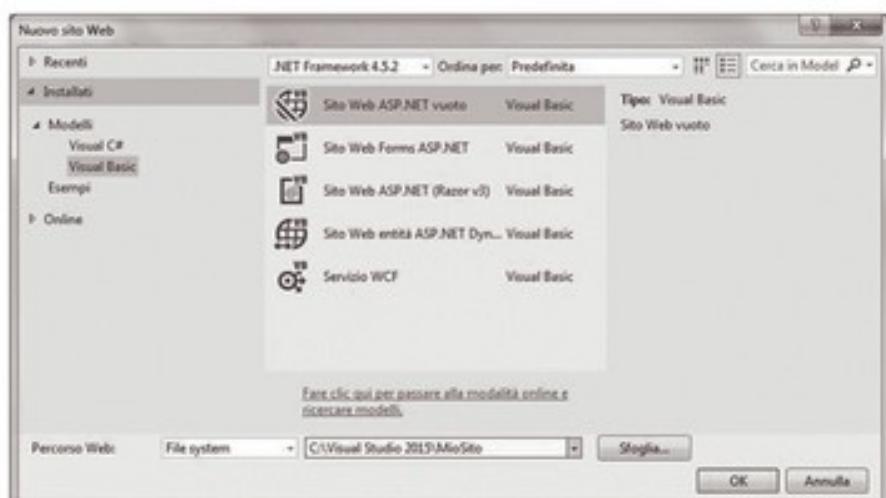
fig. 26 Creazione di un sito web con Microsoft Visual Studio



Si apre così una seconda finestra nella quale è possibile scegliere il tipo di applicazione da creare, il linguaggio da utilizzare, il percorso in cui salvare il progetto e il nome del progetto stesso.

Nell'elenco di tipi di progetto web selezionare *Sito web ASP.NET vuoto* (fig. 27):

fig. 27



Dal menu Sito Web scegliendo, Aggiungi nuovo elemento (fig. 28), compare un elenco di elementi tra i quali selezioniamo Web Form (fig. 29). Come si può notare nella fig. 29, nella casella di testo relativa al Nome si digita il nome della pagina.

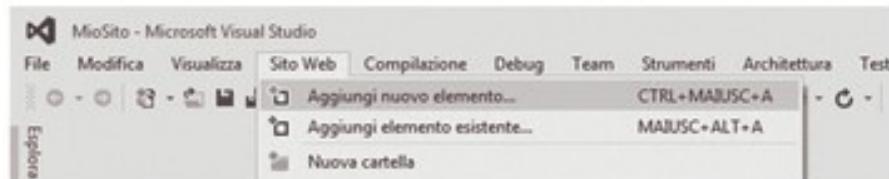


fig. 28

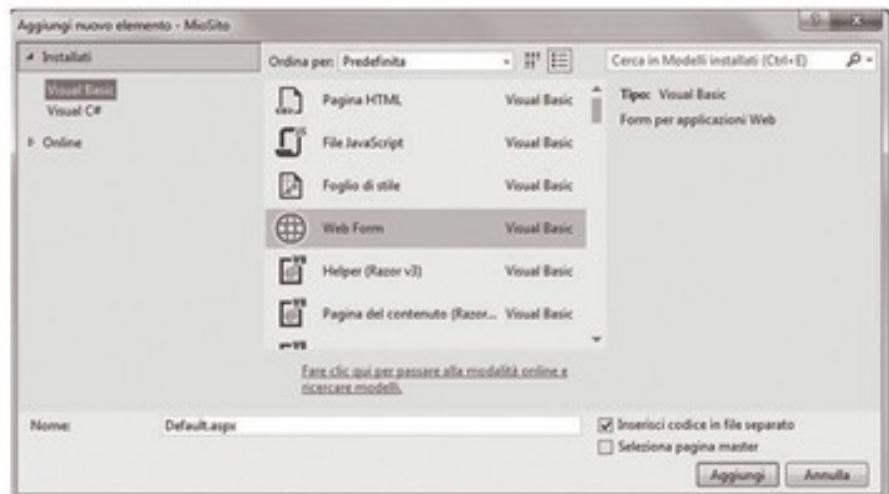


fig. 29

Ora, Visual Studio 2015 ha generato il nuovo sito web che contiene i file:

- *Default.aspx* (home page dell'applicazione web);
- *Default.aspx.vb* (che ospita il codice della pagina web);
- *Web.config* (file di configurazione) e la cartella *App_Data* (utilizzata per memorizzare file di dati, o il database, che saranno utilizzati dall'applicazione web).

Scegliendo Aggiungi compare la pagina di progettazione HTML (fig. 30) all'interno della quale si possono trascinare i controlli che interessano (label, caselle di testo, buttoni).

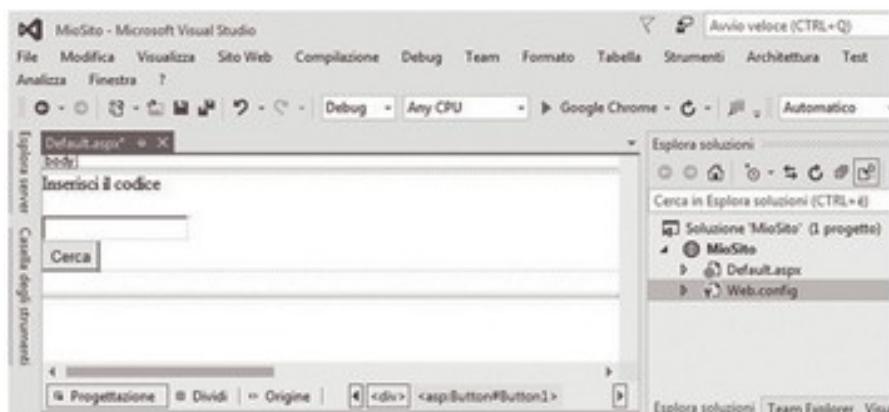


fig. 30

fig. 31

```
1 <%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default" %>
2
3 <!DOCTYPE html>
4
5 <html xmlns="http://www.w3.org/1999/xhtml">
6 <head runat="server">
7 <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
8 <title></title>
9 </head>
10 <body style="width: 1280px; height: 96px;">
11 <form id="form1" runat="server">
12 <div>
13     Inserisci il codice <br /><br />
14     <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
15     <br />
16     <asp:Button ID="btnCerca" runat="server" Text="Cerca" OnClick="btnCerca_Click"/>
17 </div>
18 </form>
19 </body>
20 </html>
```

A questo punto facendo doppio Clic sul bottone Cerca (visibile in [fig. 30](#)) possiamo scrivere il codice associato ([fig. 32](#)).

Per eseguire le pagine sul server web locale dalla pagina *Default.aspx* si clicca sul triangolo verde che avvia il browser (fig. 33).

fig. 32

```
File Modifica Visualizza Sito Web Compilazione Debug Team Strumenti Architettura Test Analizza Finestra T
  - + < > Debug Any CPU Google Chrome
Default.aspx.cs - Default.aspx
MioSito
  2 elementi
  1) Partial Class _Default
    Inherits System.Web.UI.Page
  2) elementi
    Protected Sub btnCerca_Click(sender As Object, e As EventArgs) Handles btnCerca.Click
      'Istruzioni VB.NET
    End Sub
  End Class
```

fig. 33

In fig. 34 è mostrata la home page in esecuzione.

Per chiudere l'applicazione non è sufficiente chiudere la pagina dal browser ma dalla pagina *Default.aspx* si deve cliccare sul quadratino rosso (fig. 35).

fig. 35

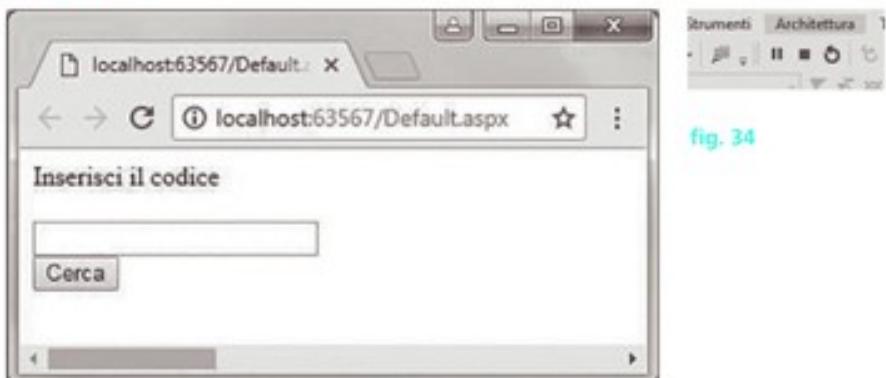


fig. 34

4. I COMANDI SQL

Istruzioni di tipo DDL (*Data Definition Language*)

ALTER TABLE <i>NomeTabella</i> ADD <i>NomeColonna</i> <i>tipo</i>	Modifica la struttura di una tabella (aggiungere colonna)
ALTER TABLE <i>NomeTabella</i> DROP <i>NomeColonna</i>	Modifica la struttura di una tabella (eliminare colonna)
CREATE DATABASE <i>NomeDatabase</i>	Crea un database
CREATE [UNIQUE] INDEX <i>NomeIndice</i> ON <i>NomeTabella</i> (<i>attributo</i> ₁ , <i>attributo</i> ₂ , ..., <i>attributo</i> _n)	Crea un indice
CREATE TABLE <i>tabella</i> (<i>NomeColonna</i> ₁ <i>tipo</i> ₁ [NOT NULL], <i>NomeColonna</i> ₂ <i>tipo</i> ₂ [NOT NULL], PRIMARY KEY (<i>chiave_primaria</i>) FOREIGN KEY (<i>chiave_esterna</i>) REFERENCES <i>NomeTabella</i> (<i>chiave</i>) ON DELETE <i>set null/no action</i> ON UPDATE <i>cascade/no action</i>)	Crea una tabella
CREATE VIEW <i>NomeVista</i> AS SELECT <i>NomeColonna</i> ₁ , <i>NomeColonna</i> ₂ ,.... FROM <i>NomeTabella</i> WHERE <i>condizione</i>	Crea una vista
DROP DATABASE <i>databasename</i>	Elimina un database
DROP INDEX <i>NomeIndice</i> ON <i>NomeTabella</i>	Elimina un indice
DROP TABLE <i>NomeTabella</i>	Elimina una tabella
DROP VIEW <i>NomeVista</i>	Elimina una vista

Istruzioni di tipo DML (*Data Manipulation Language*)

INSERT INTO <i>NomeTabella</i> [(<i>NomeColonna</i> ₁ , <i>NomeColonna</i> ₂ , ..., <i>NomeColonna</i> _n)] VALUES (<i>valore</i> ₁ , <i>valore</i> ₂ , ..., <i>valore</i> _n)	Inserisce nuovi dati in una tabella
DELETE FROM <i>NomeTabella</i> [WHERE <i>condizione</i>]	elimina i dati da una tabella secondo alcune condizioni
UPDATE <i>NomeTabella</i> SET <i>NomeColonna</i> ₁ = <i>valore</i> ₁ , SET <i>NomeColonna</i> _n = <i>valore</i> _n [WHERE <i>condizione</i>]	modifica i dati di una o più righe di una tabella

Istruzioni di tipo QL (*Query Language*)

<pre>SELECT [DISTINCT] elenco campi [INTO nuova_tabella] FROM elenco tabelle [WHERE condizione] [GROUP BY espressione gruppo] [HAVING condizione per il gruppo/risultato] [ORDERED BY campi di ordinamento [DESC]]</pre>	Seleziona dati da una o più tabelle
BETWEEN	imposta un intervallo di ricerca
DISTINCT	elimina le duplicazioni nei risultati di una query SQL
HAVING	seleziona dei gruppi creati con la clausola GROUP BY
IN/ NOT IN	imposta un insieme di ricerca
INTO	inserisce nella nuova tabella il set di risultati dell'istruzione SELECT
LIKE	effettua ricerche su dati parziali
GROUP BY	raggruppa dei dati
ORDER BY	ordina il risultato di una SELECT (DESC in ordine decrescente)
WHERE	filtra le righe da selezionare sulla base di alcune regole

Funzioni di aggregazione

COUNT(*)	numero totale di record
COUNT(campo)	numero di record con campo non nullo
SUM(campo)	somma dei valori di campo numerico
AVG(campo)	media aritmetica di campo numerico
MAX(campo)	valore massimo di campo numerico
MIN(campo)	valore minimo di campo numerico

Istruzioni di tipo DCL (*Data Control Language*)

<pre>GRANT < lista di privilegi > ON NomeTabella TO {ListaUtenti, PUBLIC}</pre>	Assegna ad un utente dei permessi specifici (SELECT, INSERT, UPDATE, DELETE)
<pre>REVOKE { ALL < lista di privilegi > } ON NomeTabella FROM {ListaUtenti, PUBLIC}</pre>	Rimuove permessi assegnati con la GRANT

Istruzioni di tipo TCL (*Transaction Control Language*)

COMMIT	rende definitive le modifiche apportate al database
ROLLBACK	annulla le modifiche apportate dopo l'ultima COMMIT

SOLUZIONI DEI TEST

1.

BASI DI DATI

Vero o falso?

1. F 2. V 3. F 4. V 5. F 6. V

Domande a scelta multipla (una sola è la risposta esatta)

7. c 8. c 9. a 10. d 11. b

2.

PROGETTARE UNA BASE DI DATI

Vero o falso?

1. F 2. F 3. V 4. V 5. F

Domande a scelta multipla (una sola è la risposta esatta)

6. c 7. c 8. d 9. a 10. d

3.

MODELLO RELAZIONALE

Vero o falso?

1. F 2. V

Domande a scelta multipla (una sola è la risposta esatta)

3. b 4. b 5. c 6. c 7. c

4.

IL LINGUAGGIO SQL

Vero o falso?

1. V 2. V 3. V 4. F 5. F

Domande a scelta multipla (una sola è la risposta esatta)

6. c 7. b 8. a 9. d 10. a

5.

PROGRAMMARE IN RETE

Vero o falso?

1. V 2. V 3. F 4. V 5. V 6. F 7. V

Domande a scelta multipla (una sola è la risposta esatta)

8. a 9. b 10. a 11. a 12. b 13. a 14. b

6.

PHP E MYSQL

Vero o falso?

1. F 2. F 3. V 4. V 5. F 6. F 7. V

Domande a scelta multipla (una sola è la risposta esatta)

8. b 9. a 10. a 11. b

7.

DATABASE IN RETE - ASP.NET

Vero o falso?

1. V 2. F 3. V 4. F 5. F 6. F 7. V 8. V

Domande a scelta multipla (una sola è la risposta esatta)

9. b 10. b 11. b

8.

BIG DATA E SISTEMI NOREL

Vero o falso?

1. F 2. V 3. F 4. V 5. V

Domande a scelta multipla (una sola è la risposta esatta)

6. b 7. c 8. d 9. a 10. d 11. a