INFO

-anche se fatto male-

Alberto Mondino

Contents

1 Basi di dati

DATO:I dati sono rappresentazioni originarie, non interpretate, di un valore.
INFORMAZIONE: L'informazione deriva da un dato, o da un insieme di dati, che sono stati contestualizzati, interpretati in modo da essere significativi.

1.1 I dati in azienda

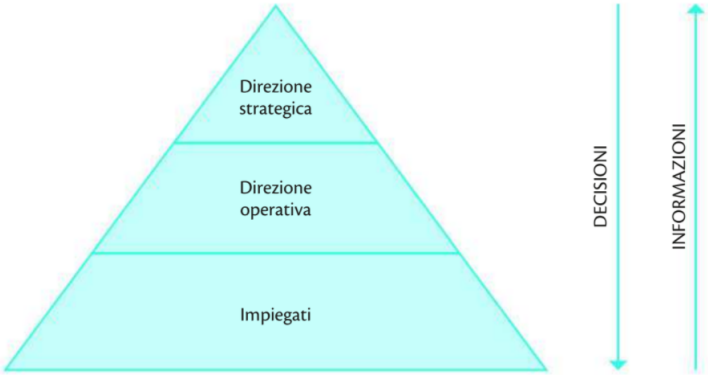
1.1.1 L'azienda

In economia il ***Know-How*** è un fattore chiave. per il successo di un'azienda
Il *Know-How* non sono altro che le conoscenze aziendali e la loro gestione.

1.1.2 Le decisioni

In azienda esistono tre decisioni:

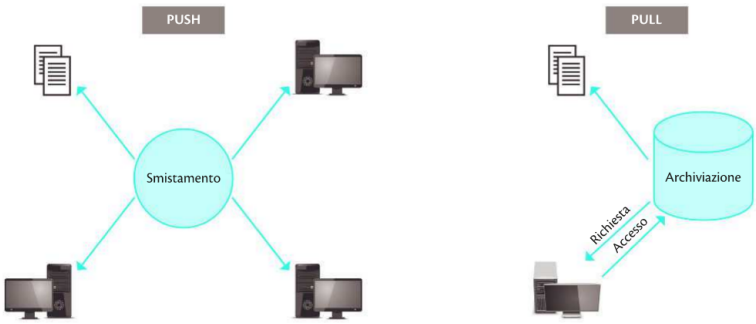
1. **Strategiche**: definiscono le linee globali dell'azienda, sono prese a livello *dirigenziale*
2. **Tattiche**: riguardano l'attuazione delle decisioni strategiche, prese dai quadri *intermedi*
3. **Operative**:riguardano la gestione delle risorse, prese dal personale *esecutivo*



1.1.3 Il flusso informativo

Il flusso informativo riguarda tutti i dati e informazioni necessari all'azienda, dati che devono arrivare dove necessario affidabilmente.
In azienda il flusso delle informazioni prevede due casi

- **Push**:l'informazione viene inviata a tutti, anche a chi non ne ha bisogno
- **Pull**: l'informazione è messa a disposizione in database accesso agli utenti

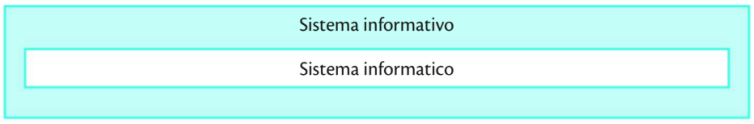


1.2 Il sistema informativo aziendale

1.2.1 Il sistema informativo

Sistema Informativo: organizza e gestisce le informazioni aziendali, si occupa delle comunicazioni dell'azienda. Il sistema informativo è formato da tutte le persone che lavorano nell'azienda e tutte le procedure necessarie per l'elaborazione dei dati.

1.2.2 Sistema informatico



Il sistema informativo automatizzato è l'insieme di tecnologie hardware e software che permettono l'automatizzazione delle funzioni informative aziendali, riducendo gli errori umani, elaborando e smistando le informazioni.

1.3 Memorizzare dati

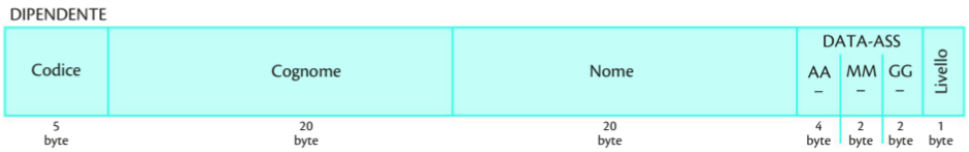
1.3.1 I File

I file consentono la conservazione dei dati, ne esistono diverse tipologie:

- **file di testo**: sequenze di caratteri ASCII, organizzate per righe
- **file binari**
- **file strutturati**: insieme di record memorizzati in codifica binaria.

1.3.2 File strutturati

Un archivio o file è un insieme di record memorizzati su un supporto di memoria permanente.



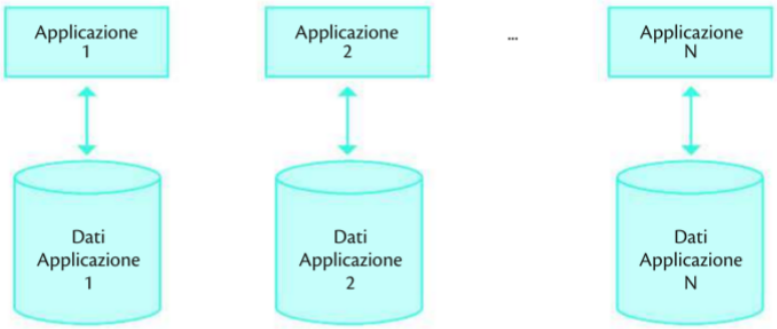
Per ottimizzare la ricerca tra i record si utilizza una **chiave primaria**, scelta tra le chiavi candidate, che identifica univocamente ciascun record. I valori assunti dalla chiave primaria devono essere tutti diversi.
Tipi di chiavi primarie:

- **chiave naturale**: chiavi che hanno un collegamento logico con il mondo (CF)
- **chiave surrogata**: chiave che non ha nessun collegae+mento logico con il mondo, ma che viene calcolata dal DBMS esclusivamente per identificare univocamente i record.

1.4 Dal file system alle basi di dati

1.4.1 Il sistema EDP

Il sistema **EDP**(**E**lectronic **D**ata **P**rocessing) è un insieme di dati memorizzati su supporti elettronici e di applicazioni informatiche. La gestione dei dati è affidata al file system.
All'interno del sistema EDP è presente un analista che ha il compito di rendere l'applicazione il più effice possibile. Nel sistema EDP ogni applicazione opera in modo indipendente dalle altre applicazioni, facendo uso dei propri dati e dei propri programmi.



1.4.2 Basi di dati e DBMS

Una base di dati è una collezione di dati strutturati, progettati per essere usati in applicazioni differenti e da utenti differenti. È un insieme di dati memorizzati senza senza duplicazioni inutili per servire più applicazioni in contemporanea, organizzati in modo da essere indipendenti dai programmi che li usano.

1.4.3 Vantaggi nell'uso dei DB

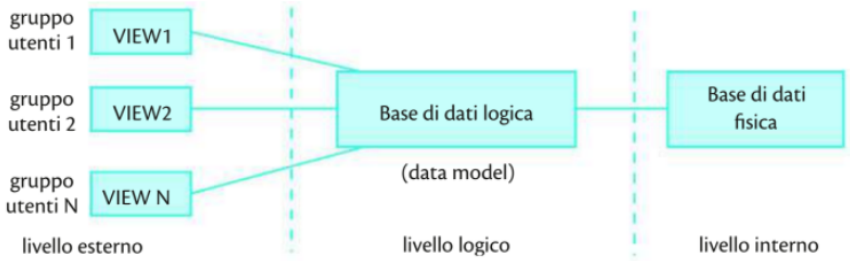
Vantaggi nell'uso dei DB rispetto ai file e rispetto a EDP:

- **Eliminazione** delle **ridondanze**, evitare duplicazioni delle informazioni
- **Eliminazione** delle **inconsistenze**, quando due dati che rappresentano la stessa informazione hanno due valori diversi
- Integrità dei dati, i dati infatti vengono protetti da cambiamenti non autorizzati o non corretti

1.5 Architettura

1.5.1 Il modello ANSI/SPARC

ANSI/SPARC è dodato di tre livelli: esterno, logico e interno, che permettono ai DBMS di nasconedere l'organizzazione dei dati.



- **Livello esterno**: detto anche livello applicativo, descrive i dati come sono visti da una o più applicazioni. Possono esistere più schemi esterni (o sottoschemi) che riguardano la "Vista" di interesse per ogni singola applicazione. Semplifica il lavoro agli utenti, mostrando loro solamente la parte che gli interessa con una rappresentazione più familiare e semplice dei dati.
- **Livello logico**: Descrive formalmente tutti gli oggetti di interesse per le applicazioni, offrendo una rappresentazione precisa e dettagliata delle strutture dati(record, campi, chiavi) necessarie per memorizzare informazioni. Permette sia l'aggiunta che la modifica di sottoschemi senza impatto sullo schema concettuale e offre uno strumento di controllo sul contenuto e sull'utilizzo del DB.
- **Livello interno**: Rappresenta la descrizione formale delle strutture fisiche degli archivi che costituiscono il DB.

1.5.2 Indipendenza logica e fisica

Per **indipendenza logica** si intende la possibilità di modificare lo schema logico senza che vada ad influenzare i programmi che si interfacciano con quella base di dati.
Per **indipendenza fisica** si intende la possibilità di modificare l'organzzazione fisica dei dati, senza dover modificare l'organizzazione logica.

1.6 Linguaggi e utenti

1.6.1 Linguaggi per i database

Per poter gestire le basi di dati tramite il DBMS si fa uso di due tipologie di linguaggi:DDL e DML.

1.6.2 DDL

DDL (**D**ata **D**efinition **L**anguage), serve per la definizione dello schema logico. Consente di definire i tipi di entità presenti nello schema concettuale e le loro relazioni. Serve anche per definire eventuali viste (sottoschemi), che consentono di selezionare solo la parte dello schema che interessa. Utilizzato principalmente dai DBA.

Istruzioni di tipo DDL (*Data Definition Language*)

ALTER TABLE *NomeTabella* ADD *NomeColonna tipo*	Modifica la struttura di una tabella (aggiungere colonna)
ALTER TABLE *NomeTabella* DROP *NomeColonna*	Modifica la struttura di una tabella (eliminare colonna)
CREATE DATABASE *NomeDatabase*	Crea un database
CREATE [UNIQUE] INDEX *NomeIndice* ON *NomeTabella* ($attributo_1$, $attributo_2$, ..., $attributo_n$)	Crea un indice
CREATE TABLE *tabella* (*NomeColonna*$_1$ *tipo*$_1$ [NOT NUL], *NomeColonna*$_2$ *tipo*$_2$ [NOT NUL], ... PRIMARY KEY (*chiave_primaria*) FOREIGN KEY (*chiave_esterna*) REFERENCES *NomeTabella*(chiave) ON DELETE *set null/no action* ON UPDATE *cascade/no action*)	Crea una tabella
CREATE VIEW *NomeVista* AS SELECT *NomeColonna*$_1$,*NomeColonna*$_2$,.... FROM *NomeTabella* WHERE *condizione*	Crea una vista
DROP DATABASE *databasename*	Elimina un database
DROP INDEX *NomeIndice* ON *NomeTabella*	Elimina un indice
DROP TABLE *NomeTabella*	Elimina una tabella
DROP VIEW *NomeVista*	Elimina una vista

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

1.6.3 DML

DML (**D**ata **M**anipulation **L**anguage), utilizzato per inserire, modificare e cancellare le informazioni contenute all'interno del DB. I DML forniscono all'utente uno strument per interrogare e modificare le informazioni contenute nel DB.

- procedurali: quando hanno gli operatori per trattare i singoli record procedurali: quando gli operatori non hanno il cocetto di posizione

Istruzioni di tipo DML (*Data Manipulation Language*)

INSERT INTO *NomeTabella* [(*NomeColonna*$_1$, *NomeColonna*$_2$, ..., *NomeColonna*$_n$)] VALUES [*valore*$_1$, *valore*$_2$, ..., *valore*$_n$]	inserisce nuovi dati in una tabella
DELETE FROM *NomeTabella* [WHERE *condizione*]	elimina i dati da una tabella secondo alcune condizioni
UPDATE *NomeTabella* SET *NomeColonna*$_1$ = *valore*$_1$ ..... SET *NomeColonna*$_n$ = *valore*$_n$ [WHERE *condizione*]	modifica i dati di una o più righe di una tabella

Istruzioni di tipo QL (*Query Language*)

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]]	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
OREDER 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*)

GRANT < *lista di privilegi* > ON *NomeTabella* TO {*ListaUtenti*, | *PUBLIC* }	Assegna ad un utente dei permessi specifici (SELECT, INSERT, UPDATE, DELETE)
REVOKE { *ALL* | < *lista di privilegi* > } ON *NomeTabella* FROM {*ListaUtenti*, | *PUBLIC*	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

1.6.4 Gli utenti

- **DBA**

  - creare e mantenere lo schema logico
  - definire e mantenere lo schema interno
  - definire e se necessario aggiornare i diritti di accesso (gran e revoke)
  - ripristinare la base di dati in caso di malfunzionamento

- **programmatori**: realizzano applicazioni utilizzando un DML.
- **utenti finali**: interagiscono col DB tramite delle maschere con interfaccia grafica

1.7 Sicurezza nelle basi di dati

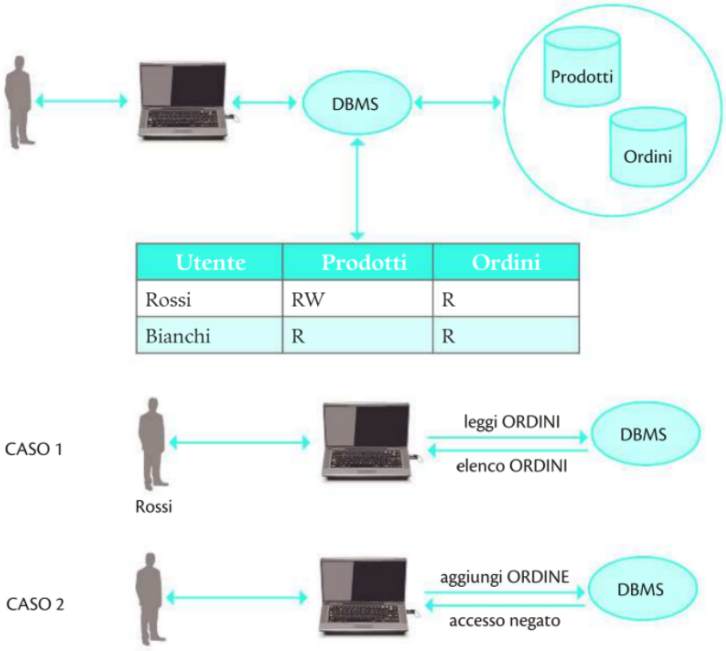
1.7.1 Problematiche di sicurezza

Le problematiche legate alla sicurezza sono di fondamentale importanza, un DBMS è sviluppato per tenere sotto controllo l'aspetto della privatezza, integrità e consistenza dei dati.

1.7.2 Privatezza

La privatezza riguarda la gestione dei permessi di accesso in un DB, è necessario garantire la possibilità a chi è autorizzato, e contemporaeamente negare l'accesso a chi non ne ha diritto.

1. Il DBA stabilisce le autorizzazioni (solo lettura, solo scrittura, ecc.) tramite il DDL. (grant e revoke)(esempio della banca)
2. Questo controllo deve essere constantemente effettuato, ad ogni richiesta il DBMS deve controllare che il richiedete abbia effettivamente i permessi per accedere ad una risorsa.



1.7.3 Vincoli di integrità

Il vincolo di integrità riguarda le relazioni esistenti tra i dati. La gestione delle informazioni deve essere sicura e corretta, ed un aspetto da controllare è la protezione contro l'aggiornamento errato del database. Per evitare questa complicazione vengono definiti dei **vincoli intra-relazionali**, che possono riguardare solo un entità (vincoli di unicità, vincoli di dominio), oppure possono definire più entità contemporaneamente.
I vincoli di integrità vengono definiti dal DBA.

1.7.4 Vincolo di integrità referenziale

Riguarda il controllo di entità che risultano correlate, anche se memorizzate su archivi diversi. Il libro fa l'esempio di un operazione su conto corrente: dobbiamo evitare che eventuali prelievi e versamenti vengano effettuati su conti correnti inesistenti.

1.7.5 Vincoli di dominio

I vincoli di dominio definiscono il campo di esistenza di un attributo. Definiscono delle regole da rispettare relative ad attributi. Ad esempio il libro fa l'esempio dell'età, infatti se stiamo definendo l'età di una persona, il vincolo sarà quello di non permettere età minori di 0.

```
Età > 0
```

oppure

```
Voto > 0 and Voto <= 10
```

1.7.6 Vincoli di relazione

Esprimono condizioni sui valori assunti tra i campi con delle relazioni.
esempi:

```
Prezzo = Costo + PercIVA*Costo
```

Il vincolo può anche essere condizionale, per cui:

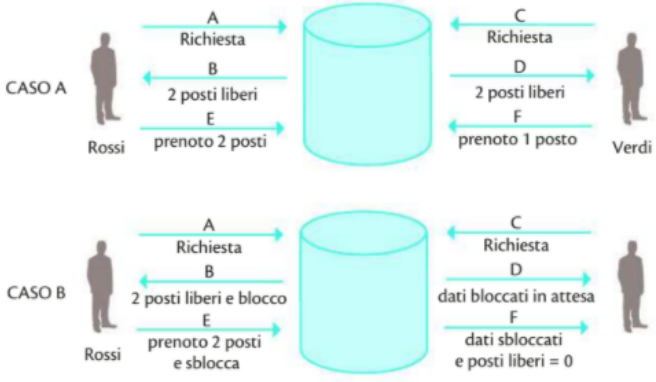
```
Tipopatente deve essere NotNull se Patente ='SI'
LODE non può esserci se il VOTO è inferiore a 30
```

1.7.7 Consistenza della base di dati

La coerenza dei dati è legata al controllo di validità che a sua volta si lega alla modifica contemporanea di dati collegati.
Una **transazione** è un insieme di istruzioni che formano un'unità di lavoro indivisibile. La transazione deve iniziare e terminare con il DB in uno stato consistente.
Una base di dati si dice *consistente* quando tutti i vincoli di validità dei dati sono rispettati.
Ogni transazione inizia con Begin_transaction e termina con un commit e l' End_transaction.
Le transazioni che non terminano correttamente vengono riportate allo stato precedente, tramite il rollback.

1.7.8 Accessi concorrenti

In una base di dati, per poter permettere la presenza di più transazioni contemporanee è necessario l'impiego di meccanismi di sincronizzazione, in modo da evitare che si verifichino errori.



2 Progettare una base di dati

2.1 La progettazione di un database

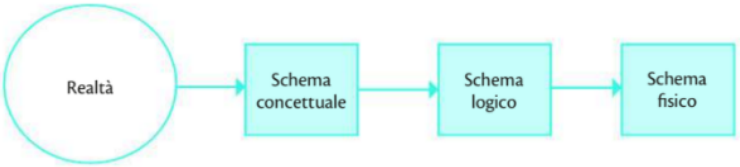
2.1.1 Dati e informazioni

Per **dato** si intende un fatto raccolto tramite osservazioni e/o misurazioni. Per **informazione** si intende l'interpretazione e il collegamento tra i Il dati.

Il modello raèèresenta una percezione di una parte della realtà,e nel processo di modellizzazione viene effettuata la selezione degli aspetti necessari per comprendere il contesto che vogliamo analizzare.

2.1.2 Fasi della progettazione

Il compito del DBA è quello di decidere quali parti della realtà sono inerenti per le applicazioni che utilizzeranno la base di dati. Normalmente la progettazione è divisa in quattro fasi:



1. **Raccolta e analisi dei dati**
   Raccolta dei requisiti del sistema informativo (dati, processi, vincoli), che vengono descritte in appositi glossari.
2. **Progettazione concettuale**
   Si formalizzano le descrizioni di come gli utenti vedono i dati (viste utenti), che vengono integrate in uno schema concettuale.
3. **Progettazione logica**
   Viene sviluppato uno schema logico sullo schema concettuale.
4. **Progettazione fisica**
   Vengono definite le strutture di memorizzaazione e i loro parametri.

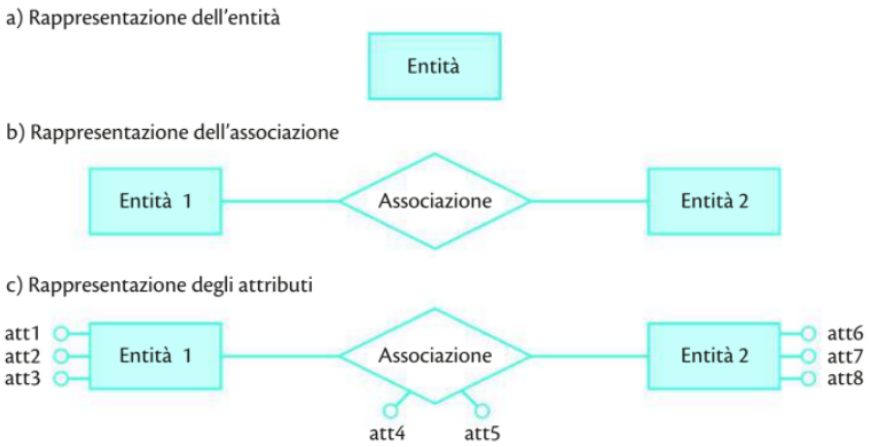
2.2 Il modello E/R - Entità e Attributi

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

- **i modelli logici**
- **i modelli concettuali**

2.2.1 Lo schema Entità/Relazioni

La descrizione concettuale dei dati si basa sull'individuazione e la definizione delle entità e delle associazioni inerenti al contesto. Tramite il modello **Entità/Relazioni** che rimanendo semplice ed intuitivo può essere descritto graficamente.



In questo modo si può descrivere la realtà tramite delle entità e le loro associazioni.

2.2.2 Entità

Le entità sono elementi dotati di caratteristiche comuni, che vengono riuniti in una classe.
Si chiama **istanza** dell'entità **un esemplare** della classe.
esempio:
Studente:
attributi:
ID:4567
Nome:Mario
Cognome:Verdi
Classe:5bi

2.2.3 Attributi

Gli attributi sono le proprietà che servono a descrivere le entità e le associazioni.
Nello schema ER vengono rappresentati con dei punti collegati alle entità o alle associazioni. E sono identificati dal tipo, dal dominio.
Possono essere:

- **Semplici**: che hanno un tipo semplice (es: Insegnante)

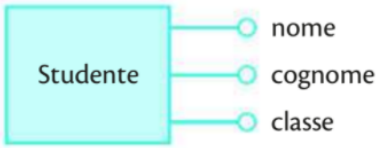


fig. 5 Attributi semplici

- **Composti**: in cui esistono sottoattributi (per esempio una data(giorno,mese,anno))

- **Multipli**: esiste un numero variabile di sottoelementi dello stesso tipo (esempio:voto scolastico).

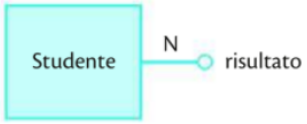


fig. 7 Attributi multipli

Gli attributi poi possono essere **opzionali**: se si permette che l'attributo possa non avere un valore assegnato (NOT NULL) o **obbligatori**: in cui è obbligatorio che l'attributo abbia un valore che non sia NOT NULL.

2.3 Le chiavi

La **chiave primaria** identifica univocamente le istanze di un entità. Nello schema ER si evidenzia dagli altri attributi perchè viene sottolineato.

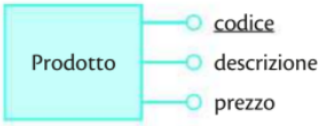


fig. 8 Chiave primaria

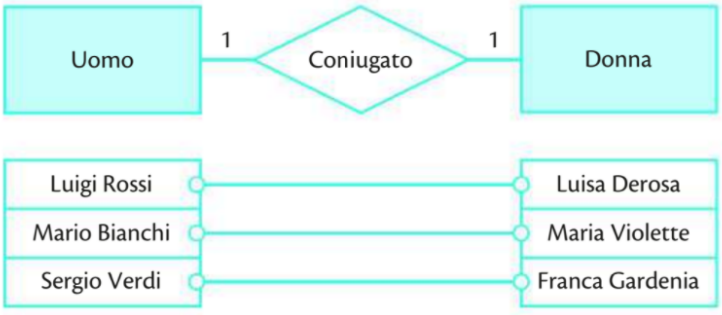
Non possono esistere chiavi primarie con valori uguali, altrimenti non si potrebbero distinguere le istanze. In un entità ci possono essere diversi attributi **candidati**, in base al contesto viene scelta la chiave primaria più attinente.

2.4 Le relazioni 1:1 e 1:N

Le relazioni sono i legami logici tra le entità. Si può parlare di classi di relazioni (1:1, 1:N, N:N).
L'associazione viene rappresentata graficamente con un rombo che collega le entità associate.

2.4.1 Tipi di relazioni tra entità

- **Relazioni 1:1**: ad ogni elemento del primo insieme ne corrisponde uno e uno solo del secondo insieme.



- **Relazioni 1:N**: A un elemento del primo insieme possono corrispondere più elementi del secondo insieme, ma non viceversa.

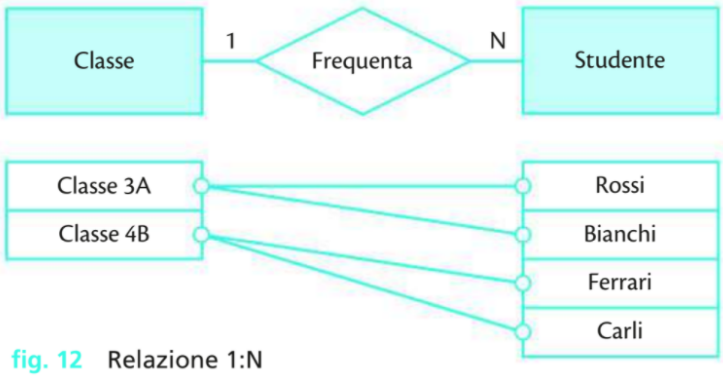
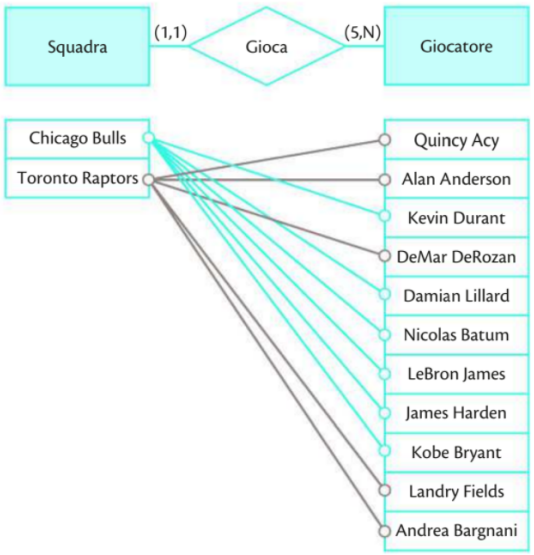


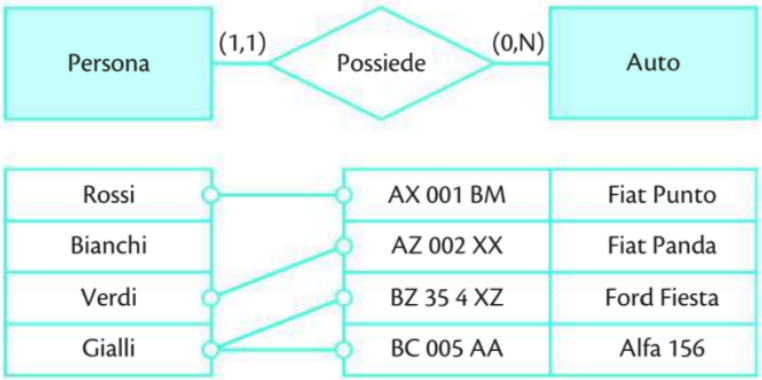
fig. 12 Relazione 1:N

2.5 Cardinalità delle associazioni

La cardinalità di un associazione definisce il numero di istanze con cui può partecipare un associazione. E' identificata da due numeri, che indicano la cardinalità **minima** e **massima**.



La cardinalità minima uguale a zero indica che l'entità partecipa in **modo opzionale** all'associazione.



2.6 Le relazioni con N:N e le relazioni con gli attributi

Le relazioni N:N riguardano le associazioni che coinvolgono N elementi, sia della prima che della seconda entità

2.6.1 Associazioni com gli attributi

3 Il linguaggio SQL

3.1 Definire lo schema

SQL(**S**tructured **Q**uery **L**anguage) è un linguaggio che ti permette di eseguire operazioni CRUD (Create, Read, Update, Delete) in basi di dati relazionali. SQL fa parte è un linguaggio ***non procedurale*** in cui non devi specificare come vuoi trovare qualcosa, ma solamente cosa vuoi trovare.
Il linguaggio SQL ha diverse categorie specializzate:

- **DDL** (**D**ata **D**efinition **L**anguage): creare e modificare schemi di database
- **DML** (**D**ata **M**anipulation **L**anguage): inserire, modificare e gestire dati
- **DQL** (**D**ata **Q**uery **L**anguage): interrogare i dati
- **TCL** (**T**ransaction **C**ontrol **L**anguage): creare e gestire transazioni

3.1.1 Creazione di database

Per creare un database si usa la seguente sintassi:
CREATE DATABASE nome_database
Per eliminare un database esistente si usa il comando:
DROP DATABASE nome_database

3.1.2 Creazione di tabelle

Per creare una tabella si usa il comando CREATE TABLE:

```
CREATE TABLE nome-tabella
  (nome-colonna1 tipo1 {NOT NULL}
  .....
  nome-colonnaN tipoN)
```

Accanto al nome di ogni tabella deve essere specificato il tipo di dato. Se necessario possono essere specificate anche altre clausole:

- **NOT NULL** (il valore non può essere nullo)
- **PRIMARY KEY** (definisce la chiave primaria)
- **UNIQUE** (dice che il campo non può avere valori duplicati)
- **FOREIGN KEY** (definisce una chiave esterna)

3.1.3 Tipi di dati

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

in caso espandi

3.1.4 Creare un indice

Per migliorare l'efficienza di ricerca delle righe nelle tabelle si definisce un indice sul campo di ricerca.
Un indice su una tabella esistente si crea con l'istruzione CREATE INDEX

```
CREATE UNIQUE INDEX nome-indice
    ON nome-tabella(nome-colonna)
```

Gli indici sono importantissimi per velocizzare le operazioni di interrogazione. Quando si aggiornano le tabelle si deve ricontrollare la validità degli indici.

3.2 Modificare lo schema di una base di dati

Lo schema dei dati può essere modificato sia eliminando tabelle o indici, sia modificando la struttura stessa della tabella.

3.2.1 ALTER TABLE

L'istruzione ALTER TABLE permette di modificare a 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 esistente:

```
ALTER TABLE identificatore-tabella
ADD nome-colonna tipo
```

esempio:

```
ALTER TABLE Giocatori
ADD Nazione CHAR(6)
```

Per **cancellare una colonna** da una tabella già esistente si usa:

```
ALTER TABLE identificatore-tabella
DROP nome-colonna
```

esempio:

```
ALTER TABLE Giocatori
DROP Nazione
```

Per **cambiare il tipo di una colonna** si fa:

```
ALTER TABLE identificatore-tabella
CHANGE nome-colonna nome-colonna nuovotipo
```

esempio:

4 Interrogazioni

4.1 Domande professoressa

1. Informazioni all'interno aziende?
2. Come si arriva alla necessità delle basi di dati?
3. Integrità e consistenza delle basi di dati?
4. Il modello ANSI/SPARC?
5. Quali sono le tipologie di utenti di un DBMS?
6. Quali sono i vincoli delle basi di dati?
7. Con che istruzione l'amministratore della base di dati assegna i permessi agli utenti?
8. Il modello logico relazionale per le basi di dati?
9. Nello schema logico relazionale, da dove arrivano queste tabelle? che caratteristiche hanno i dati che ci sono dentro?
10. Che caratteristiche che devono avere i dati nelle tabelle relazionali?
11. Fai un esempio di una 1:N
12. Cos'è JDBC?
13. Cos'è il modello relazionale per le basi di dati?
14. Cos'è SQL?
15. Cosa sono e quali sono le cardinalità?
16. Come si fa a togliere i permessi in un DB?
17. Gli attributi nella progettazione ER vengono solo dalle entità?
18. Cosa fanno le chiavi primarie?
19. Cosa sono le chiavi candidate?
20. Cosa sono le chiavi surrogate?
21. Che tipi di attributi esistono? fai degli esempi
22. Cosa si intende per dominio?
23. Cos'è la ristrutturazione? fai degli esempi
24. Quali sono le fasi della progettazione concettuale? analizza ogni fase
25. Come sono tradotte le associazioni in tabelle? Cosa differenzia la loro tradizione dalle istanze?
26. Che vincoli devono rispettare le tabelle in un sistema relazionale?
27. SQL- com'è strutturata la gerarchia delle Queary?
28. Che cosa sono gli operatori aggregati? Quali sono? Dove si possono utilizzare?
29. Qual è la differenza tra JOIN left e JOIN right?
30. Quali sono i due conceti chiave delle Queary (connessione e statement)

4.2 Consigli della prof

- Parlare lentamente
- Non andare a infilarti in cose che non conosci
- Non divagare
- Quando parli di qualcosa fai esempi