

Sistemi informativi

▼ 1.0 - Introduzione ai sistemi informativi

▼ 1.1 - Sistemi informativi e basi di dati

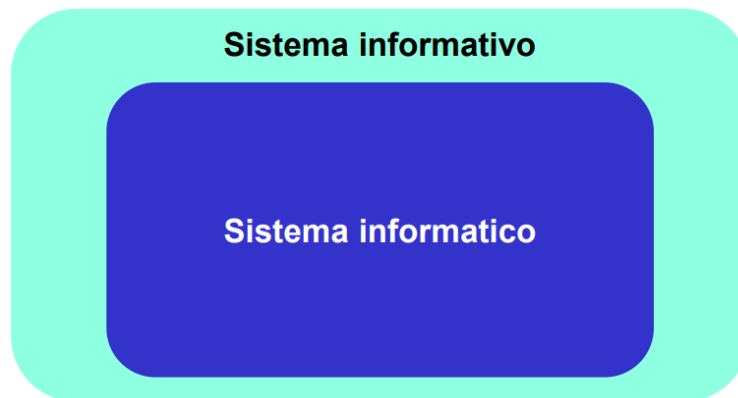
Sistemi informativi

Un **sistema informativo** è un componente di un'organizzazione (azienda, ente, ecc.) il cui scopo è gestire le informazioni utili per gli scopi di tale organizzazione.

L'informazione è una risorsa molto importante, a volte anche più o meno importante del capitale, in quanto se lavorata nella maniera corretta porta all'elaborazione di scelte e, nel caso di aziende, di indicazioni strategiche. Per creare un'informazione di valore occorre interpretare nella maniera corretta i dati grezzi ed inserirli nel giusto contesto.

Sistema informativo e sistema informatico

Un sistema informativo non fa necessariamente utilizzo di strumenti automatici propri dell'informatica, ma, nel caso in cui esista, la parte automatizzata di un sistema informativo viene detta **sistema informatico**.



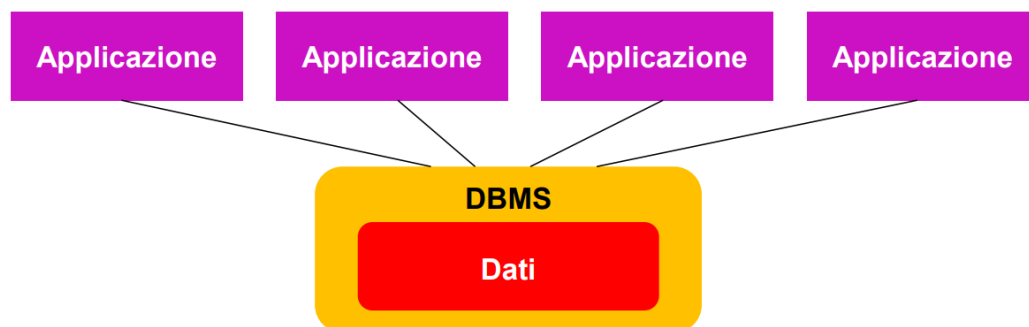
Relazione tra sistema informativo e sistema informatico.

Basi di dati

Un sistema informatico gestisce solitamente le informazioni tramite l'utilizzo di una rappresentazione codificata dei dati di interesse, chiamata base di dati o database.

In termini tecnici, una **base di dati** è una collezione di dati gestita da un **DBMS** (Data Base Management System).

Un DBMS è un sistema software in grado di gestire collezioni di dati condivise da più applicazioni e utenti.



Un DBMS, come vedremo in seguito più nello specifico, è in grado di gestire grandi quantità di dati, garantirne la persistenza ed elevate prestazioni, e di offrire una visione strutturata dei dati dipendente dal modello logico supportato.

Per eseguire operazioni su un database è necessario scrivere istruzioni in un linguaggio supportato dal DBMS.

I RDBMS sono DBMS che supportano il modello relazionale dei dati.

Esistono **3 ruoli** principali che vengono ricoperti da chi lavora con i data base:

- **Utente**

Fa utilizzo del database conoscendone il modello dei dati, i linguaggi supportati dal DBMS e le modalità con cui un'applicazione può collegarsi ad esso.

- **Progettista**

Realizza la struttura del database comprendendo come i requisiti informativi di un'organizzazione possano tradursi in una struttura

concreta.

- **Amministratore**

Amministra il database avendo conoscenze su come è fatto il DBMS, soprattutto per motivi di efficienza.

In questo corso verranno trattati i ruoli dell'utente e del progettista.

▼ 1.2 - DBMS

Definizione di DBMS

Un **DBMS** è un sistema software che gestisce una grande quantità di dati, persistenti e condivisi.

Per funzionare al meglio un DBMS deve ottimizzare la sua efficienza, garantire l'affidabilità dei suoi dati anche a seguito di errori o con parti non funzionanti, controllare gli accessi tramite autorizzazioni e controllare la concorrenza.

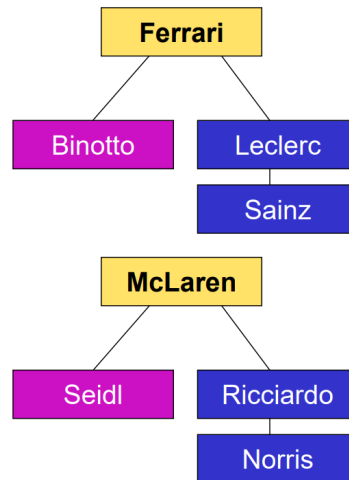
Solitamente i DBMS in commercio presentano anche ulteriori funzionalità per semplificare la descrizione dei dati e lo sviluppo di applicazioni.

Il modello dei dati

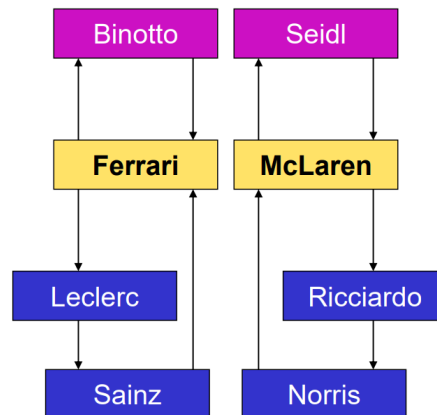
Un **modello dei dati** è una collezione di concetti che vengono utilizzati per descrivere i dati, le loro associazioni e i vincoli che devono rispettare.

Quando viene definito un modello dei dati viene solitamente descritto anche il suo meccanismo di struttura. Tali meccanismi di struttura differiscono in base al tipo di organizzazione dei dati (ad albero, a grafo ecc.).

I modelli dei dati più utilizzati sono il modello gerarchico, quello reticolare e quello relazionale:



Modello gerarchico.



Modello reticolare.

| | | | |
|---------|---------|---------|-----------|
| Ferrari | Binotto | Ferrari | Leclerc |
| McLaren | Seidl | Ferrari | Sainz |
| | | McLaren | Ricciardo |
| | | McLaren | Norris |

Modello relazionale.

Schemi e istanze

In ogni database si hanno due componenti:

- Lo **schema**, che descrive la struttura dei dati, e rappresenta la parte intensionale del database.
- L'**istanza**, ovvero i dati veri e propri del database, che rappresenta la parte estensionale del database.

Lo schema ha lo scopo di interpretare e dare un senso ai dati dell'istanza.

Mentre l'istanza varia nel tempo, lo schema tende a rimanere invariato.

Cataloghi

La descrizione degli schemi di un database è memorizzata nel database stesso, all'interno dei **cataloghi**, i quali descrivono le tabelle, i vincoli sui dati, le autorizzazioni ecc.

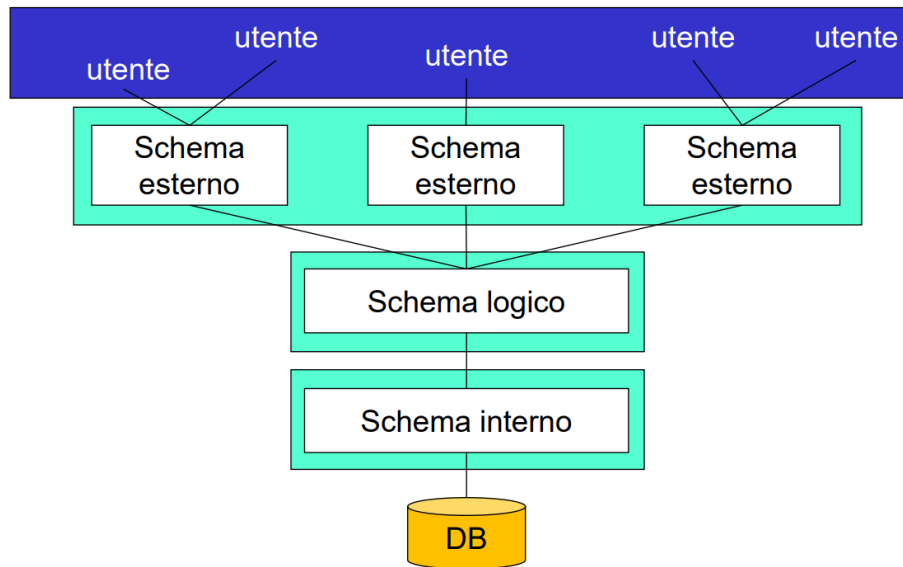
Tramite l'utilizzo dei cataloghi si riesce dunque a mantenere una descrizione dei dati centralizzata ed essendo i cataloghi memorizzati nel database anche questi possono essere interrogati come avviene per i dati.

Architettura a 3 livelli di un DBMS

Tra gli obiettivi di un DBMS vi sono quelli di fornire:

- **Indipendenza fisica**, in quanto l'organizzazione fisica dei dati non deve comportare effetti collaterali sul funzionamento degli applicativi che li usano.
- **Indipendenza logica**, in quanto non è utile o conveniente che ogni utente abbia la stessa visione uniforme dell'organizzazione dei dati.

Per garantire queste due caratteristiche viene utilizzata un'organizzazione dei DBMS detta architettura a 3 livelli



Architettura a 3 livelli di un DBMS.

Livello fisico (o interno)

Il **livello fisico** di un DBMS consiste nell'insieme di file, memorizzati all'interno di memorie, che contengono i dati, gli indici e le informazioni utili per il corretto funzionamento del database.

La gestione del DB fisico è a carico di chi amministra il DBMS e non degli utenti, che quindi possono concentrarsi su aspetti di più alto livello per lo sviluppo delle loro applicazioni.

| | | | |
|---------|-------|-----------|--------|
| Ferrari | Sainz | McLaren | Norris |
| Binotto | | Seidl | |
| Leclerc | | Ricciardo | |

File: c://apps/DB/data/scuderie

Livello fisico.

Livello logico

Il livello logico consiste in una serie di strutture, nel modello relazionale le relazioni, che non dipendono dal livello fisico, ma forniscono un significato ai dati presenti in tale livello.

A fronte di una ristrutturazione nel livello fisico, il livello logico rimane invariato.

| Manager | | Piloti | |
|----------|---------|----------|-----------|
| Scuderia | TeamMgr | Scuderia | Pilota |
| Ferrari | Binotto | Ferrari | Leclerc |
| McLaren | Seidl | Ferrari | Sainz |
| | | McLaren | Ricciardo |
| | | McLaren | Norris |

Livello logico.

Livello delle viste (o esterno)

Il livello esterno è costruito a partire dal livello logico tramite delle viste che descrivono parte dello schema logico a seconda delle esigenze dell'utente.

Una vista fornisce una visione personalizzata del database combinando dati provenienti da relazioni anche diverse, consentendo in questo modo il controllo degli accessi mascherando all'utente dei dati riservati, oppure anche calcolando in maniera dinamica dei nuovi dati ricavabili da quelli già presenti nel database.

| Piloti | | Il 28/09/2021 | EtàPiloti | |
|---------|-------------|---------------|-----------|-----|
| Pilota | DataNascita | | Pilota | Età |
| Leclerc | 16/10/1997 | ➔ | Leclerc | 23 |
| Sainz | 01/09/1994 | | Sainz | 27 |

Vista.

| Manager | | Piloti | | MgrPiloti | |
|----------|---------|----------|-----------|-----------|---------|
| Scuderia | TeamMgr | Scuderia | Pilota | Pilota | TeamMgr |
| Ferrari | Binotto | Ferrari | Leclerc | Leclerc | Binotto |
| McLaren | Seidl | Ferrari | Sainz | Sainz | Binotto |
| | | McLaren | Ricciardo | Ricciardo | Seidl |
| | | McLaren | Norris | Norris | Seidl |

Livello esterno.

▼ 2.0 - Il modello relazionale

▼ 2.1 - Relazioni

Relazionale, gerarchico e reticolare

Il **modello relazionale** venne introdotto in seguito all'introduzione dei modelli gerarchico e reticolare, creando uno standard che enfatizza la semplicità d'uso rispetto all'efficienza, la quale rende la struttura della base di dati molto più complicata.

La principale differenza del modello relazionale rispetto ai due modelli antecedenti risiede principalmente nel modo in cui vengono rappresentati i legami, in quanto nel modello relazionale vengono solamente utilizzati valori, a differenza dei puntatori utilizzati nel gerarchico e nel reticolare. Inoltre, il modello relazionale è il primo ad essere formalmente definito tramite una teoria relazionale utile per la progettazione di database, la definizione di linguaggi e l'ottimizzazione delle richieste.

Relazione matematica

Per introdurre il concetto di relazione per quanto riguarda le basi di dati è utile rivedere la definizione matematica di relazione.

Si considerino n insiemi D_1, \dots, D_n non necessariamente distinti, il prodotto cartesiano $D_1 \times \dots \times D_n$ è l'insieme di tutte le n -ple (d_1, \dots, d_n) tali che $d_1 \in D_1, \dots, d_n \in D_n$.

Una relazione matematica è un qualunque sottoinsieme del prodotto cartesiano

$D_1 \times \dots \times D_n$.

- D_1, \dots, D_n sono i **domini** della relazione.
- Il valore di n è detto **grado** della relazione.
- Il numero di n -ple della relazione è detto **cardinalità**.

Relazione nel modello relazionale

La nozione di relazione viene utilizzata nel modello relazionale nel seguente modo:

$\text{partite} \subseteq \text{String} \times \text{String} \times \text{Integer} \times \text{Integer}$

| | | | |
|-------------|----------------|-----|----|
| Benetton TV | Poliform Cantù | 100 | 71 |
| Kinder BO | MontePaschi SI | 90 | 51 |
| Paf BO | Adr RM | 62 | 97 |
| Adr RM | Kinder BO | 80 | 62 |

Notiamo che la posizione e l'interpretazione di ciascun dominio è fondamentale. Inoltre, nel modello relazionale, ad ogni dominio della relazione viene associato un nome che ne specifica il ruolo, chiamato **attributo**.

| TeamCasa | TeamOspite | PuntiCasa | PuntiOspite |
|-------------|----------------|-----------|-------------|
| Benetton TV | Poliform Cantù | 100 | 71 |
| Kinder BO | MontePaschi SI | 90 | 51 |
| Paf BO | Adr RM | 62 | 97 |
| Adr RM | Kinder BO | 80 | 62 |

Attributi nel modello relazionale.

Tramite l'utilizzo degli attributi la struttura non è più posizionale, ovvero l'ordine degli attributi non ha più rilevanza, in quanto ogni attributo è univoco.

Una relazione può dunque essere definita informalmente come una tabella in cui le colonne, ovvero gli attributi, rappresentano le proprietà di interesse, mentre le righe (tuple) rappresentano ciascuna uno specifico oggetto presente nel database. Dunque quando si parla di relazione ci si riferisce ad un oggetto composto da due parti:

- Lo **schema**, formato dal nome della relazione e dagli attributi.

Studenti

| Matricola | Cognome | Nome | DataNascita | Email |
|-----------|---------|------|-------------|-------|
|-----------|---------|------|-------------|-------|

- L'**istanza**, formata dai dati veri e propri (può essere vuota).

| | | | | |
|-------|---------|---------|------------|------------------------|
| 29323 | Bianchi | Giorgio | 21/06/1978 | gbianchi@alma.unibo.it |
| 35467 | Rossi | Anna | 13/04/1978 | anna.rossi@yahoo.it |
| 39654 | Verdi | Marco | 20/09/1979 | mverdi@mv.com |
| 42132 | Neri | Lucia | 15/02/1978 | lucia78@cs.ucsd.edu |

Notazione di base

Per denotare un insieme di attributi si usa la notazione semplificata:

- A in luogo di $\{A\}$ e XY in luogo di $X \cup Y$.
- ABC o A, B, C al posto di $\{A, B, C\}$.

Se t è una tupla su X e $A \in X$, allora $t[A]$ è $t.A$ è il valore di t su A . La stessa notazione si usa per insiemi di attributi, e denota una tupla: $t[A, B]$ è una tupla su $\{A, B\}$.

Per riferirsi all'istanza della relazione con schema $R(X)$ si usa r , ovvero il nome in minuscolo della relazione.

Partite(TeamCasa,TeamOspite,PuntiCasa,PuntiOspite)

partite =

| | | | |
|-------------|----------------|-----|----|
| Benetton TV | Poliform Cantù | 100 | 71 |
| Kinder BO | MontePaschi SI | 90 | 51 |
| Paf BO | Adr RM | 62 | 97 |
| Adr RM | Kinder BO | 80 | 62 |

Lo schema di un database relazionale è un insieme di schemi di relazioni con nomi distinti:

$$R = \{R_1(X_1), \dots, R_n(X_n)\} \quad (R_i \neq R_j, \forall i \neq j)$$

L'istanza di un database con schema $R = \{R_1(X_1), \dots, R_n(X_n)\}$ è un insieme di istanze di relazioni:

$$r = \{r_1, \dots, r_n\}$$

▼ 2.2 - Vincoli

Vincoli di integrità dei dati

Una relazione non rappresenta un contenitore di dati arbitrari, ma occorre specificare quali sono i **vincoli** che le istanze contenute in essa devono rispettare affinché possano essere considerate valide.

1NF, solo domini atomici


Il primo vincolo fornito dal modello relazionale è quello di non poter utilizzare domini strutturati (array, set, liste, alberi ecc.) per la definizione delle relazioni. Si possono dunque utilizzare solamente domini atomici, ovvero non ulteriormente decomponibili, i quali si dicono anche in **prima forma normale** (1NF).

Partendo dalle informazioni originali, per ottenere relazioni in 1NF occorre effettuare un'attività di **normalizzazione dei dati**.

| | | |
|-----------------------------------|---|-------|
| Ricevuta n. 231 del 12/02/2002 | | |
| Coperti | 2 | 3,00 |
| Antipasti | 1 | 5,80 |
| Primi | 2 | 11,45 |
| Secondi | 2 | 22,30 |
| Caffè | 2 | 2,20 |
| Vino | 1 | 8,00 |
| Totale (Euro) | | 52,75 |

Ricevute

Dettaglio



| Numero | Data | Totale |
|--------|------------|--------|
| 231 | 12/02/2002 | 52,75 |
| 352 | 13/02/2002 | ... |
| ... | ... | ... |

| Numero | Quantità | Descrizione | Prezzo |
|--------|----------|-------------|--------|
| 231 | 2 | Coperti | 3,00 |
| 231 | 1 | Antipasti | 5,80 |
| 231 | 2 | Primi | 11,45 |
| 231 | 2 | Secondi | 22,30 |
| 231 | 2 | Caffè | 2,20 |
| 231 | 1 | Vino | 8,00 |
| 352 | 1 | Coperti | 1,50 |

Normalizzazione dei dati di ricevute.

Valori nulli

In alcuni casi le informazioni che si vogliono rappresentare non corrispondono esattamente allo schema relazionale prescelto, ad esempio può accadere che per alcune tuple potrebbe non essere possibile specificare un valore nel caso in cui non lo si conosca.

In questi casi occorre fare utilizzo del valore nullo **NULL**, il quale denota l'assenza di un valore nel dominio. Questa pratica è più sicura dell'utilizzo di un valore speciale del dominio (0, -1, "" ecc.), in quanto tale valore potrebbe diventare significativo in futuro, e le applicazioni dovrebbero conoscere il significato di tale valore.

$$t[A] \in \text{dom}(A) \cup \{\text{NULL}\}$$

Se due tuple hanno entrambe valore NULL per lo stesso attributo, non si può dire che esse abbiano lo stesso valore per tale attributo, ovvero:

$$\text{NULL} \neq \text{NULL}$$

Vincoli di dominio

La correttezza sintattica di un'istanza non è condizione sufficiente affinché i dati rappresentino un'informazione possibile nel contesto considerato.

Uno dei vincoli che può essere necessario dover stabilire è quello di **dominio**, il quale restringe i possibili valori utilizzati per uno specifico attributo ad un determinato range del dominio.

Esami

| Matricola | CodCorso | Voto | Lode |
|-----------|----------|------|-------|
| 29323 | 483 | 28 | NO |
| 39654 | 729 | 30 | Sì |
| 29323 | 913 | 31 | NO |
| 35467 | 913 | 30 | FORSE |

- Il Voto deve essere compreso tra 18 e 30
(Voto ≥ 18) AND (Voto ≤ 30)
- La Lode può solo assumere i valori `Sì` o `NO`
(Lode = `Sì`) OR (Lode = `NO`)

Esempio di vincoli di dominio.

Vincoli di tupla

I **vincoli di tupla** esprimono condizioni che ogni tupla deve rispettare.

Esami

| Matricola | CodCorso | Voto | Lode |
|-----------|----------|------|------|
| 29323 | 483 | 28 | NO |
| 39654 | 729 | 30 | Sì |
| 29323 | 913 | 26 | Sì |
| 35467 | 913 | 30 | NO |

- La Lode si può assegnare solo se il Voto è 30:
(Voto = 30) OR NOT(Lode = `Sì`)

Esempio di vincolo di tupla.

Vincoli di chiave

I **vincoli di chiave** vietano la presenza di tuple distinte che possiedono lo stesso valore per uno o più attributi.

▼ 2.3 - Chiavi

Chiavi e superchiavi

Dato uno schema $R(X)$, un insieme di attributi $K \subseteq X$ è:

- Una **superchiave** se e solo se in ogni istanza ammissibile r di $R(X)$ non esistono due tuple distinte t_1 e t_2 tali che $t_1[K] = t_2[K]$.
- Una **chiave** se e solo se è una superchiave minimale, ovvero non esiste $K' \subset K$ con K' superchiave.

Esempio nella relazione studenti:

- $\{\text{Matricola}\}$ e $\{\text{CodiceFiscale}\}$ sono due chiavi.
- $\{\text{Matricola}, \text{Cognome}\}$ è una superchiave.
- $\{\text{Cognome}, \text{Nome}, \text{DataNascita}\}$ non è una superchiave.

Le chiavi sono lo strumento principale attraverso il quale vengono correlati i dati in relazioni diverse. Se due tuple hanno un valore nullo in una chiave è difficile sapere se si riferisca alla stessa istanza presente in un'altra tupla, per questo vengono utilizzate delle **chiavi primarie**, sulle quali non si ammettono valori nulli.

Foreign key

Si considerino due schemi $R_1(X_1)$ e $R_2(X_2)$ di un database R e sia $Y \subseteq X_2$, allora l'insieme Y si dice **foreign key** se l'insieme dei valori di Y nell'istanza r_2 è un sottoinsieme dei valori della chiave primaria di $R_1(X_1)$ presenti nell'istanza r_1 .

Un esempio di foreign key è il seguente:

| | | | | | |
|----------|------------------|---------------------|---------|-------------|------|
| Studenti | <u>Matricola</u> | Cognome | Nome | DataNascita | |
| | 29323 | Bianchi | Giorgio | 21/06/1978 | |
| | 35467 | Rossi | Anna | 13/04/1978 | |
| | 39654 | Verdi | Marco | 20/09/1979 | |
| | 42132 | Neri | Lucia | 15/02/1978 | |
| Corsi | <u>CodCorso</u> | Titolo | | Docente | Anno |
| | 483 | Analisi | | Biondi | 1 |
| | 729 | Analisi | | Neri | 1 |
| | 913 | Sistemi Informativi | | Castani | 2 |
| Esami | <u>Matricola</u> | <u>CodCorso</u> | Voto | Lode | |
| | 29323 | 483 | 28 | NO | |
| | 39654 | 729 | 30 | Sì | |
| | 29323 | 913 | 26 | NO | |
| | 35467 | 913 | 30 | NO | |

In Esami, {Matricola} e {CodCorso} sono foreign key.

Supponendo che, in Esami, {Matricola} non sia una foreign key, allora è possibile avere un'istanza del tipo:

| | | | | |
|----------|------------------|-----------------|---------|-------------|
| Studenti | <u>Matricola</u> | Cognome | Nome | DataNascita |
| | 29323 | Bianchi | Giorgio | 21/06/1978 |
| | 35467 | Rossi | Anna | 13/04/1978 |
| | 39654 | Verdi | Marco | 20/09/1979 |
| Esami | <u>Matricola</u> | <u>CodCorso</u> | Voto | Lode |
| | 29323 | 483 | 28 | NO |
| | 39654 | 729 | 30 | Sì |
| | 41235 | 913 | 26 | NO |
| | 35467 | 913 | 30 | NO |



La foreign key e la primary key possono includere attributi con nomi diversi.

| | | | | |
|-------|---------------------|-----------------|---------|------|
| Corsi | <u>Codice</u> | Titolo | Docente | Anno |
| | 483 | Analisi | Biondi | 1 |
| | 729 | Analisi | Neri | 1 |
| Esami | <u>NumMatricola</u> | <u>CodCorso</u> | Voto | Lode |
| | 29323 | 483 | 28 | NO |

La foreign key e la primary key possono anche far parte della stessa relazione.

| | | | | |
|-----------|---------------|-------------|-----|-----------------|
| Personale | <u>Codice</u> | Nome | ... | CodResponsabile |
| | 123 | Mario Rossi | ... | 325 |
| | 134 | Gino Verdi | ... | 325 |
| | 325 | Anna Neri | ... | ... |

▼ 3.0 - Algebra relazionale

▼ 3.1 - Operatori

Un **linguaggio di manipolazione** permette di interrogare e modificare istanze di basi di dati.

A parte i linguaggi utente come SQL, esistono linguaggi formalmente definiti che enfatizzano aspetti essenziali dell'interazione con un database relazionale. Uno di questi è quello dell'**algebra relazionale**, il quale è costituito da un insieme di operatori che si applicano a una o più relazioni e che producono una relazione.

Operatori di base **unari**: selezione, proiezione e ridenominazione.

Operatori di base **binari**: join, unione e differenza.

Selezione

L'operazione di **selezione**, σ , permette di selezionare un sottoinsieme delle tuple di una relazione, applicando a ciascuna di esse una formula booleana.

$$\sigma_F(r) = \{t \mid t \in r \wedge F(t)\}$$

Esami

| Matricola | CodCorso | Voto | Lode |
|-----------|----------|------|------|
| 29323 | 483 | 28 | NO |
| 39654 | 729 | 30 | Sì |
| 29323 | 913 | 26 | NO |
| 35467 | 913 | 30 | NO |
| 31283 | 729 | 30 | NO |

$\sigma_{(\text{Voto} = 30) \text{ AND } (\text{Lode} = \text{'NO'})}(\text{Esami})$

| Matricola | CodCorso | Voto | Lode |
|-----------|----------|------|------|
| 35467 | 913 | 30 | NO |
| 31283 | 729 | 30 | NO |

$\sigma_{(\text{CodCorso} = 729) \text{ OR } (\text{Voto} = 30)}(\text{Esami})$

| Matricola | CodCorso | Voto | Lode |
|-----------|----------|------|------|
| 39654 | 729 | 30 | Sì |
| 35467 | 913 | 30 | NO |
| 31283 | 729 | 30 | NO |

Proiezione

L'operazione di **proiezione**, π , permette di selezionare un sottoinsieme degli attributi di una relazione. Essendo un insieme i valori duplicati vengono eliminati.

$$\pi_Y(r) = \{t[Y] \mid t \in r\}$$

| Corsi | CodCorso | Titolo | Docente | Anno |
|-------|----------|---------------------|---------|------|
| | 483 | Analisi | Biondi | 1 |
| | 729 | Analisi | Neri | 1 |
| | 913 | Sistemi Informativi | Castani | 2 |

$\pi_{\text{CodCorso, Docente}}(\text{Corsi})$

| CodCorso | Docente |
|----------|---------|
| 483 | Biondi |
| 729 | Neri |
| 913 | Castani |

Join naturale

L'operazione di **join naturale**, $\triangleright\triangleleft$, combina le tuple di due relazioni diverse sulla base dell'uguaglianza, detta match, dei valori degli attributi comuni alle due relazioni.

$$r_1 \triangleright\triangleleft r_2 = \{t \mid t[X_1] \in r_1 \wedge t[X_2] \in r_2\}$$

Esami

| Matricola | CodCorso | Voto | Lode |
|-----------|----------|------|------|
| 29323 | 483 | 28 | NO |
| 39654 | 729 | 30 | Sì |
| 29323 | 913 | 26 | NO |
| 35467 | 913 | 30 | NO |

Corsi

| CodCorso | Titolo | Docente | Anno |
|----------|---------------------|---------|------|
| 483 | Analisi | Biondi | 1 |
| 729 | Analisi | Neri | 1 |
| 913 | Sistemi Informativi | Castani | 2 |

Esami $\triangleright\triangleleft$ Corsi

| Matricola | CodCorso | Voto | Lode | Titolo | Docente | Anno |
|-----------|----------|------|------|---------------------|---------|------|
| 29323 | 483 | 28 | NO | Analisi | Biondi | 1 |
| 39654 | 729 | 30 | Sì | Analisi | Neri | 1 |
| 29323 | 913 | 26 | NO | Sistemi Informativi | Castani | 2 |
| 35467 | 913 | 30 | NO | Sistemi Informativi | Castani | 2 |

È possibile che una tupla di una relazione non faccia match con nessuna altra tupla dell'altra relazione, in questo caso tale tupla viene detta

dangling.

Osservazioni

- Per il join naturale esistono 2 casi limiti, uno nel quale nessuna tupla fa match, dunque il risultato è vuoto, e l'altro nel quale ogni tupla di r_1 si combina con ogni tupla di r_2 . Ne segue che la **cardinalità** del join naturale ($|r_1 \bowtie r_2|$) è compresa tra i seguenti valori:

$$0 \leq |r_1 \bowtie r_2| \leq |r_1| \cdot |r_2|$$

- Se il join è eseguito su una superchiave di $R_1(X_1)$, allora ogni tupla di r_2 fa match con al massimo una tupla di r_1 , quindi $|r_1 \bowtie r_2| \leq |r_2|$.
- Se il join viene eseguito su una chiave che è chiave primaria in $R_1(X_1)$ e foreign-key in $R_2(X_2)$, allora $|r_1 \bowtie r_2| = |r_2|$.
- Quando tra le due relazioni del join non ci sono attributi in comune ($X_1 \cap X_2 = \emptyset$), allora due tuple fanno sempre match, per cui tale join corrisponde al prodotto cartesiano non ordinato tra le tuple delle due relazioni.

Unione

L'operazione di **unione**, \cup , fornisce l'insieme delle tuple appartenenti almeno ad una delle due relazioni date in input.

$$r_1 \cup r_2 = \{t \mid t \in r_1 \vee t \in r_2\}$$

Differenza

L'operazione di **differenza**, $-$, fornisce l'insieme delle tuple appartenenti alla prima relazione data in input e non alla seconda.

$$r_1 - r_2 = \{t \mid t \in r_1 \wedge t \notin r_2\}$$

Ridenominazione

L'operazione di **ridenominazione**, ρ , modifica lo schema di una relazione, cambiando i nomi di uno o più attributi.

| Redditi | | $\rho_{\text{CodiceFiscale} \leftarrow \text{CF}}(\text{Redditi})$ | |
|--------------|------------|--|------------|
| CF | Imponibile | CodiceFiscale | Imponibile |
| BNCGRG78F21A | 10000 | BNCGRG78F21A | 10000 |

Self-join

È possibile utilizzare l'operazione di ridenominazione per effettuare il self-join, ovvero il join di una relazione con sè stessa in maniera sensata.

Genitori

| Genitore | Figlio |
|----------|--------|
| Luca | Anna |
| Maria | Anna |
| Giorgio | Luca |
| Silvia | Maria |
| Enzo | Maria |

Per trovare nonni e nipoti:

$\rho_{\text{Nonno, Genitore} \leftarrow \text{Genitore, Figlio}}(\text{Genitori})$

| Nonno | Genitore |
|---------|----------|
| Luca | Anna |
| Maria | Anna |
| Giorgio | Luca |
| Silvia | Maria |
| Enzo | Maria |

$\rho_{\text{Nonno, Genitore} \leftarrow \text{Genitore, Figlio}}(\text{Genitori}) \bowtie \text{Genitori}$

| Nonno | Genitore | Figlio |
|---------|----------|--------|
| Giorgio | Luca | Anna |
| Silvia | Maria | Anna |
| Enzo | Maria | Anna |

... poi si può ridenominare Figlio in Nipote e proiettare su {Nonno, Nipote}

Operatori derivati

Gli operatori sinora visti definiscono completamente l'algebra relazionale. Tuttavia per praticità è utile definire altri operatori derivati, quali la divisione e il theta-join.

Divisione

L'operazione di **divisione**, \div , di r_1 per r_2 , con r_1 su $R_1(X_1 X_2)$ e r_2 su $R_2(X_2)$, è l'insieme di tuple con schema X_1 tale che, facendo il prodotto cartesiano con r_2 , ciò che si ottiene è una relazione contenuta in r_1 .

$$r_1 \div r_2 = \{t \mid \{t\} \bowtie r_2 \subseteq r_1\}$$

Voli

| Codice | Data |
|--------|------------|
| AZ427 | 21/07/2001 |
| AZ427 | 23/07/2001 |
| AZ427 | 24/07/2001 |
| TW056 | 21/07/2001 |
| TW056 | 24/07/2001 |
| TW056 | 25/07/2001 |

Linee

| Codice |
|--------|
| AZ427 |
| TW056 |

Voli ÷ Linee

| Data |
|------------|
| 21/07/2001 |
| 24/07/2001 |

(Voli ÷ Linee) ▷◁ Linee

| Codice | Data |
|--------|------------|
| AZ427 | 21/07/2001 |
| AZ427 | 24/07/2001 |
| TW056 | 21/07/2001 |
| TW056 | 24/07/2001 |

La divisione trova le date con voli per tutte le linee

La divisione trova le date con voli per **tutte** le linee

Theta-join

Il **theta-join** è la combinazione di prodotto cartesiano e selezione.

$$r_1 \bowtie_F r_2 = \sigma_F(r_1 \bowtie r_2)$$

Con r_1 e r_2 senza attributi in comune e F composta di predicati di join, ovvero del tipo $A \theta B$, con $A \in X_1$ $A \in X_1$ $B \in X_2$.

Ricercatori

| Nome | CodProgetto |
|---------|-------------|
| Rossi | HK27 |
| Verdi | HAL2000 |
| Bianchi | HK27 |
| Verdi | HK28 |
| Neri | HAL2000 |

Progetti

| Sigla | Responsabile |
|---------|--------------|
| HK27 | Bianchi |
| HAL2000 | Neri |
| HK28 | Verdi |

Ricercatori >< CodProgetto=Sigla Progetti

| Nome | CodProgetto | Sigla | Responsabile |
|---------|-------------|---------|--------------|
| Rossi | HK27 | HK27 | Bianchi |
| Verdi | HAL2000 | HAL2000 | Neri |
| Bianchi | HK27 | HK27 | Bianchi |
| Verdi | HK28 | HK28 | Verdi |
| Neri | HAL2000 | HAL2000 | Neri |

Ricercatori >< (CodProgetto=Sigla) AND Progetti
(Nome ≠ Responsabile)

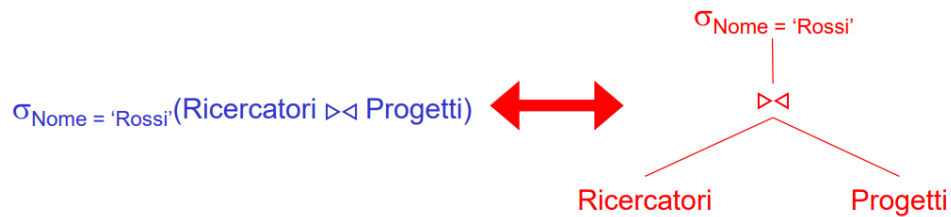
| Nome | CodProgetto | Sigla | Responsabile |
|-------|-------------|---------|--------------|
| Rossi | HK27 | HK27 | Bianchi |
| Verdi | HAL2000 | HAL2000 | Neri |

▼ 3.2 - Espressioni

Espressioni

Gli operatori dell'algebra relazionale si possono combinare tra loro in **espressioni**.

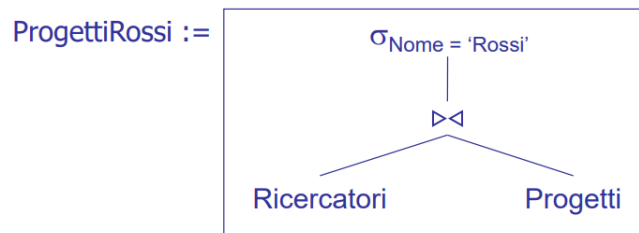
Le espressioni in algebra relazionale si possono visualizzare sia tramite rappresentazioni lineari che tramite rappresentazioni grafiche ad albero. In quest'ultimo caso la valutazione delle operazioni procede bottom-up.



Viste

In algebra relazionale è possibile assegnare un nome alle espressioni, definendo in questo modo delle **viste**.

L'utilizzo di viste semplifica la scrittura di espressioni complesse.



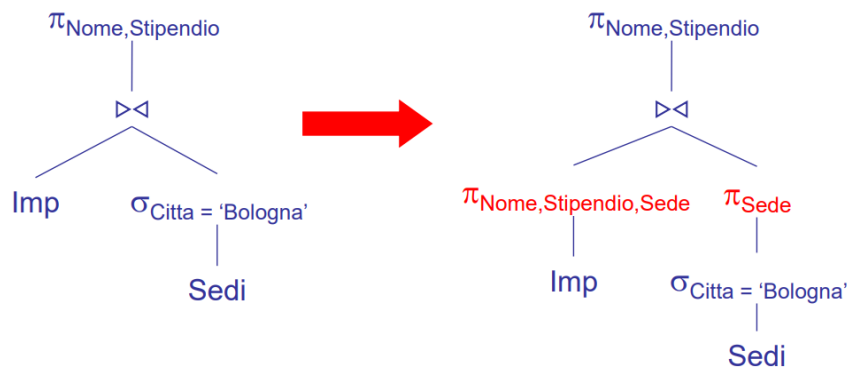
Equivalenza di espressioni

Due espressioni $E1$ e $E2$ si dicono **equivalenti** rispetto a DB e si scrive $E1 \equiv_{DB} E2$ se e solo se per ogni istanza db di DB producono lo stesso risultato, $E1(db) = E2(db)$.

In alcuni casi l'equivalenza non dipende da un DB specifico ma è generale, in quel caso si scrive $E1 \equiv E2$.

Due espressioni equivalenti garantiscono lo stesso risultato, ma ciò non significa che la scelta sia indifferente in termini di risorse necessarie. In fase di **ottimizzazione** infatti si utilizzano regole di equivalenza che consentano di trasformare un'espressione in una equivalente ma valutabili in modo più efficiente.

Una delle tecniche utilizzate per migliorare le prestazioni di un'espressione è quella di eliminare il prima possibile gli attributi che non servono più.



Regole di equivalenza

Le principali regole di equivalenza nell'algebra relazionale sono le seguenti:

- Il join naturale è commutativo e associativo.

$$E_1 \bowtie E_2 \equiv E_2 \bowtie E_1 \quad (E_1 \bowtie E_2) \bowtie E_3 \equiv E_1 (E_2 \bowtie E_3)$$

- Selezione e proiezione si possono raggruppare.

$$\sigma_{F1}(\sigma_{F2}(E)) \equiv \sigma_{F1 \wedge F2}(E) \quad \pi_Y(\pi_{YZ}(E)) \equiv \pi_Y(E)$$

- Selezione e proiezione commutano (F si riferisce solo ad attributi di Y).

$$\pi_Y(\sigma_F(E)) \equiv \sigma_F(\pi_Y(E))$$

- Push-down della selezione rispetto al join (F è sullo schema di E_1).

$$\sigma_F(E_1 \bowtie E_2) \equiv (\sigma_F E_1) \bowtie E_2$$

Dimostrazione

Occorre dimostrare che $\sigma_F(E_1 \bowtie E_2) \equiv (\sigma_F E_1) \bowtie E_2$, ovvero che:

$$\begin{aligned} \sigma_F(E_1 \bowtie E_2) &\subseteq (\sigma_F E_1) \bowtie E_2 \\ &\quad \wedge \\ (\sigma_F E_1) \bowtie E_2 &\subseteq \sigma_F(E_1 \bowtie E_2) \end{aligned}$$

- $\sigma_F(E_1 \bowtie E_2) \subseteq (\sigma_F E_1) \bowtie E_2$

Dobbiamo dunque dimostrare che $t \in \sigma_F(E_1 \bowtie E_2) \implies t \in (\sigma_F E_1) \bowtie E_2$.

Per ipotesi abbiamo che $t \in \sigma_F(E_1 \bowtie E_2)$.

Per definizione di selezione possiamo concludere che $F(t)$ è vera e che $t \in E_1 \bowtie E_2$. Per definizione di join naturale abbiamo quindi che, essendo X_1 e X_2 gli attributi nello schema di E_1 e di E_2 , $t[X_1] \in E_1$ e $t[X_2] \in E_2$.

Sapendo per ipotesi che $F(t)$ è vera per ogni attributo di t e che $t[X_1] \in E_1$, allora in particolare sappiamo che $t[X_1] \in \sigma_F(E_1)$ e che quindi, visto che $t[X_2] \in E_2$, $t[X_1] \in \sigma_F(E_1 \bowtie E_2)$.

- $(\sigma_F E_1) \bowtie E_2 \subseteq \sigma_F(E_1 \bowtie E_2)$

Il ragionamento è analogo.

▼ 3.3 - Valori nulli

La presenza di valori nulli nelle istanze richiede un'estensione della semantica degli operatori. Va premesso che esistono diversi approcci nel trattamento delle istanze con valori nulli nelle espressioni, l'approccio che qui utilizziamo è quello tradizionale, analogo a quello utilizzato in SQL e dai DBMS tradizionali.

Proiezione, unione e differenza con valori nulli

Le operazioni di proiezione, unione e differenza continuano a comportarsi usualmente anche con valori nulli, dunque due tuple sono uguali anche se ci sono dei NULL.

Impiegati

| Cod | Nome | Ufficio |
|-----|-------|---------|
| 123 | Rossi | A12 |
| 231 | Verdi | NULL |
| 373 | Verdi | A27 |
| 435 | Verdi | NULL |

$\pi_{\text{Nome, Ufficio}}(\text{Impiegati})$

| Nome | Ufficio |
|-------|---------|
| Rossi | A12 |
| Verdi | NULL |
| Verdi | A27 |

Responsabili

| Cod | Nome | Ufficio |
|------|-------|---------|
| 123 | Rossi | A12 |
| NULL | NULL | A27 |
| 435 | Verdi | NULL |

$\text{Impiegati} \cup \text{Responsabili}$

| Cod | Nome | Ufficio |
|------|-------|---------|
| 123 | Rossi | A12 |
| 231 | Verdi | NULL |
| 373 | Verdi | A27 |
| 435 | Verdi | NULL |
| NULL | NULL | A27 |

Selezione con valori nulli

Per lavorare esplicitamente con i valori nulli nell'operazione di selezione si introduce l'operatore di confronto IS (es. A IS NULL, A IS NOT NULL).

Impiegati

| Cod | Nome | Ufficio |
|-----|-------|---------|
| 123 | Rossi | A12 |
| 231 | Verdi | NULL |
| 373 | Verdi | A27 |
| 385 | NULL | A27 |

$\sigma_{(\text{Ufficio} = \text{'A27'}) \text{ AND } (\text{Nome} = \text{'Verdi'})}(\text{Impiegati})$

| Cod | Nome | Ufficio |
|-----|-------|---------|
| 373 | Verdi | A27 |

$\sigma_{\text{Ufficio} = \text{'A12'}}(\text{Impiegati})$

| Cod | Nome | Ufficio |
|-----|-------|---------|
| 123 | Rossi | A12 |

$\sigma_{(\text{Ufficio} = \text{'A27'}) \text{ OR } (\text{Nome} = \text{'Verdi'})}(\text{Impiegati})$

| Cod | Nome | Ufficio |
|-----|-------|---------|
| 231 | Verdi | NULL |
| 373 | Verdi | A27 |
| 385 | NULL | A27 |

$\sigma_{(\text{Ufficio} = \text{'A12'}) \text{ OR } (\text{Ufficio} \neq \text{'A12'})}(\text{Impiegati})$

| Cod | Nome | Ufficio |
|-----|-------|---------|
| 123 | Rossi | A12 |
| 373 | Verdi | A27 |
| 385 | NULL | A27 |

$\sigma_{\text{Ufficio IS NULL}}(\text{Impiegati})$

| Cod | Nome | Ufficio |
|-----|-------|---------|
| 231 | Verdi | NULL |

$\sigma_{(\text{Ufficio IS NULL}) \text{ AND } (\text{Nome IS NULL})}(\text{Impiegati})$

| Cod | Nome | Ufficio |
|-----|------|---------|
|-----|------|---------|

Join naturale con valori nulli

Il join naturale non combina due tuple se queste hanno entrambe valore NULL su un attributo in comune.

| Impiegati | Cod | Nome | Ufficio |
|-----------|-----|-------|---------|
| | 123 | Rossi | A12 |
| | 231 | Verdi | NULL |
| | 373 | Verdi | A27 |
| | 435 | Verdi | NULL |

| Responsabili | Ufficio | Cod |
|--------------|---------|------|
| | A12 | 123 |
| | A27 | NULL |
| | NULL | 231 |

Impiegati \bowtie Responsabili

| Cod | Nome | Ufficio |
|-----|-------|---------|
| 123 | Rossi | A12 |

Intersezione con valori nulli

In assenza di valori nulli l'intersezione si può esprimere nei seguenti due modi:

- $r_1 \cap r_2 = r_1 \bowtie r_2$.
- $r_1 \cap r_2 = r_1 - (r_1 - r_2)$.

In presenza di valori nulli si ha che:

- $r_1 \bowtie r_2$ non contiene tuple con valori nulli.
- $r_1 - (r_1 - r_2)$ contiene tuple con valori nulli.

Outer-join

L'**outer-join** svolge il lavoro di un join naturale con la differenza che le tuple dangling, ovvero le tuple che non fanno match con nessuna tupla dell'altra relazione, vengono inserite nel risultato completandole con valori nulli.

Esistono 3 varianti dell'outer-join:

- **Left** ($= \triangleright \triangleleft$): solo le tuple dangling dell'operando sinistro vengono completate con valori nulli.
- **Right** ($\triangleright \triangleleft =$): solo le tuple dangling dell'operando destro vengono completate con valori nulli.
- **Full** ($= \triangleright \triangleleft =$): le tuple dangling di entrambe gli operandi vengono completate con valori nulli.

Ricercatori

| Nome | CodProgetto |
|---------|-------------|
| Rossi | HK27 |
| Bianchi | HK27 |
| Verdi | HK28 |

Progetti

| CodProgetto | Responsabile |
|-------------|--------------|
| HK27 | Bianchi |
| HAL2000 | Neri |

Ricercatori $=\triangleright \triangleleft$ Progetti

| Nome | CodProgetto | Responsabile |
|---------|-------------|--------------|
| Rossi | HK27 | Bianchi |
| Bianchi | HK27 | Bianchi |
| Verdi | HK28 | NULL |

Ricercatori $=\triangleright \triangleleft =$ Progetti

| Nome | CodProgetto | Responsabile |
|---------|-------------|--------------|
| Rossi | HK27 | Bianchi |
| Bianchi | HK27 | Bianchi |
| Verdi | HK28 | NULL |
| NULL | HAL2000 | Neri |

Ricercatori $\triangleright \triangleleft =$ Progetti

| Nome | CodProgetto | Responsabile |
|---------|-------------|--------------|
| Rossi | HK27 | Bianchi |
| Bianchi | HK27 | Bianchi |
| NULL | HAL2000 | Neri |

▼ 4.0 - Linguaggio SQL

Il **linguaggio SQL** è un linguaggio standard per DBMS relazionali, che riunisce in sé funzionalità di DDL (Data Definition Language), DML (Data Manipulation Language) e DCL (Data Control Language).

Tale linguaggio è **dichiarativo**, ovvero non specifica la sequenza delle operazioni da compiere per ottenere il risultato, e **relazionalmente completo**, ossia ogni espressione dell'algebra relazionale può essere tradotta in SQL.

Il modello dei dati in SQL è basato su **tabelle** anziché relazioni, e questo causa il fatto che possono essere presenti più righe, o tuple, uguali.

Data Definition Language (DDL)

Il **DDL** di SQL permette di definire schemi di relazioni, anche detti tabelle, modificarli ed eliminarli. Consente inoltre di stabilire vincoli, creare nuovi domini oltre a quelli predefiniti e di creare nuove viste, ovvero tabelle virtuali.

Data Manipulation Language (DML)

Il **DML** di SQL permette, oltre alla possibilità di aggiungere, modificare ed eliminare tuple dalle tabelle del DBMS, di effettuare tutte le operazioni viste in precedenza nella sezione di algebra relazionale, tramite le 4 istruzioni seguenti:

- **SELECT**
- **INSERT**
- **DELETE**
- **UPDATE**

L'interpretazione semantica di una query SQL non va intesa come l'effettiva modalità di esecuzione di tale operazione, in quanto ogni DBMS ha le proprie strategie per determinare la modalità di esecuzione più efficiente, sfruttando le regole di equivalenza dell'algebra relazionale.

▼ 5.0 - Modello Entity-Relationship

Il **modello Entity-Relationship** è uno standard de facto per la progettazione concettuale, ovvero uno schema che rappresenti la realtà di interesse in maniera indipendente dal DBMS.

Esso si posiziona come un modello di astrazione intermedio tra il modello logico da rappresentare e ciò che viene creato tramite il DBMS.

Di seguito verranno presentati le principali componenti del modello Entity-Relationship.

Entità

Un'**entità** consiste in un insieme di oggetti della realtà di interesse che possiedono caratteristiche comuni e che esistono autonomamente.

Graficamente un'entità si rappresenta tramite un rettangolo.



L'**istanza** di un'entità è uno specifico oggetto appartenente a tale entità.

Associazioni

Un'**associazione** consiste in un legame logico tra più entità.

Graficamente un'associazione si rappresenta con un rombo.



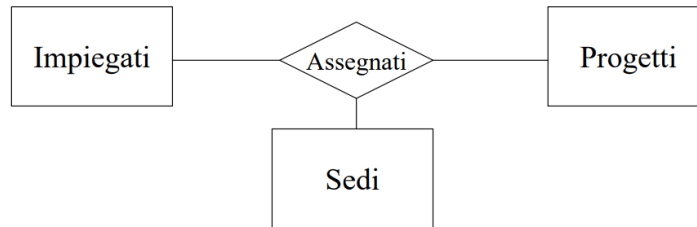
Considerando l'esempio in figura, se p è un'istanza di Persone e c un'istanza di Città, la coppia (p, c) è un'istanza dell'associazione Risiedono.

Il **grado dell'associazione** indica il numero di istanze di entità coinvolte in un'istanza dell'associazione.

- associazione binaria: grado = 2

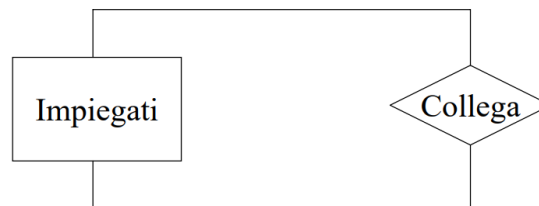


- associazione ternaria: grado = 3



Grado di un'associazione.

È possibile stabilire più associazioni tra le stesse entità e persino creare associazioni ad anello, ossia associazioni che coinvolgono più volte la stessa entità e che dunque mettono in relazione tra loro le istanze appartenenti ad una stessa entità.

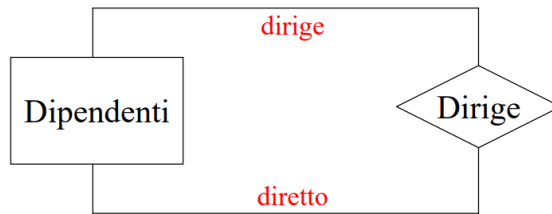


Associazione ad anello.

Un'associazione ad anello può essere o meno:

- Simmetrica: $(a, b) \in A \implies (b, a) \in A$.
- Riflessiva: $(a, a) \in A$.
- Transitiva: $(a, b) \in A \wedge (b, c) \in A \implies (a, c) \in A$.

Nelle associazioni ad anello non simmetriche occorre specificare, per ogni ramo dell'associazione, il relativo ruolo.

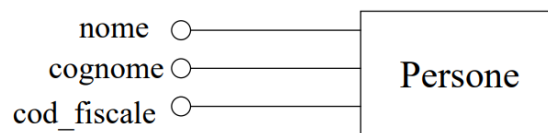


Associazione ad anello non simmetrica.

Attributi

Un'**attributo** è una proprietà elementare di un'entità o di un'associazione.

Graficamente viene rappresentato con un pallino connesso all'entità/associazione di appartenenza.

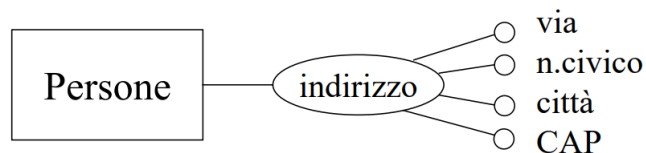


Attributi.

Ogni attributo è definito su un dominio di valori, quindi esso associa ad ogni istanza di entità/associazione un valore del corrispondente dominio.

Attributi composti

Un **attributo composto** consiste in un attributi che si ottiene aggregando altri attributi, i quali presentano una forte affinità nel loro uso e significato e che vengono detti sotto-attributi.



Attributo composto.

Il **dominio** di un attributo composto corrisponde al prodotto cartesiano dei domini dei suoi sotto-attributi.

Vincoli di cardinalità

È possibile specificare per ogni attributo il suo **vincolo di cardinalità**, ovvero una coppia (min-card, max-card) che specifica il numero minimo di attributi di tale tipologia possono essere associati ad un'istanza dell'entità/associazione.

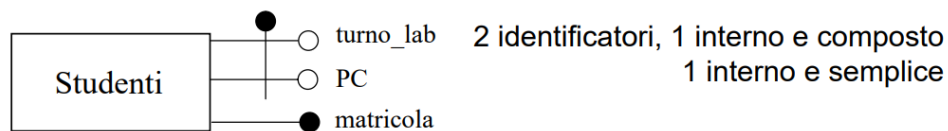
Identificatori

Un'**identificatore** permette l'individuazione univoca delle istanze di un'entità ed è minimale, dunque nessun sottoinsieme proprio di un identificatore può a sua volta essere un identificatore.

Ogni entità deve avere almeno un'identificatore e può averne anche più di 1.

Un'identificatore di un'entità può essere **interno** ed **esterno**.

Se il numero di elementi che costituiscono l'identificatore è pari a 1 si parla di identificatore **semplice**, altrimenti **composto**.



Esempio di identificatore.

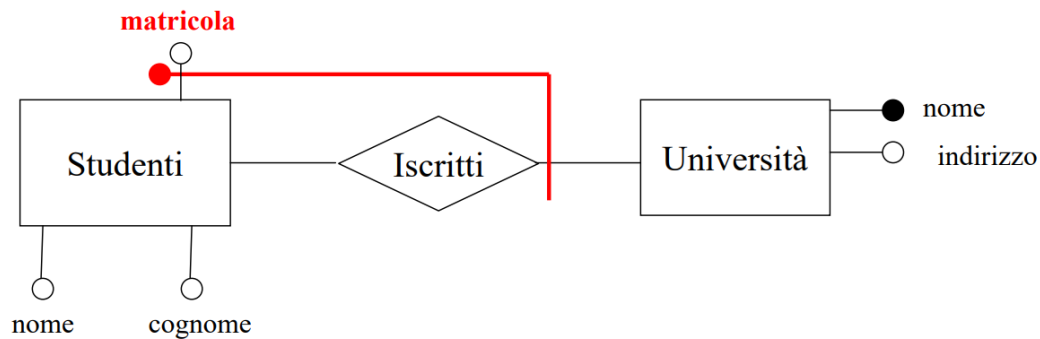
Identificatori esterni

Un'**identificatore esterno** consente di identificare un'entità E tramite altre entità, collegate ad E tramite associazioni, più eventuali attributi di E (in questo caso vengono prende il nome di identificatore misto).

Tali tipologie di identificatori servono a gestire quelle situazioni che nella realtà sono molto comuni in cui un'istanza di un'entità ha i valori di alcuni attributi

che sono univoci solamente rispetto alle altre istanze che fanno parte di un certo contesto.

Vediamo ad esempio il caso dell'identificatore matricola per l'entità studenti. Tale identificare identifica uno studente solo all'interno della sua università, ma non in tutte le università del mondo, dunque occorre utilizzare un'identificare esterno per rappresentare questo vincolo.



Identificatore esterno.

Vincoli di cardinalità

I **vincoli di cardinalità** sono coppie di valori (min-card, max-card) che, per ogni entità che partecipa ad un'associazione, specifica il numero minimo e massimo di istanze dell'associazione a cui un'istanza dell'entità può partecipare.



- $\text{min-card}(\text{Automobili}, \text{Proprietà}) = 0$: esistono automobili non possedute da alcuna persona
- $\text{max-card}(\text{Automobili}, \text{Proprietà}) = 1$: ogni automobile può avere al più un proprietario
- $\text{min-card}(\text{Persone}, \text{Proprietà}) = 0$: esistono persone che non posseggono alcuna automobile
- $\text{max-card}(\text{Persone}, \text{Proprietà}) = n$: ogni persona può essere proprietaria di un numero arbitrario di automobili

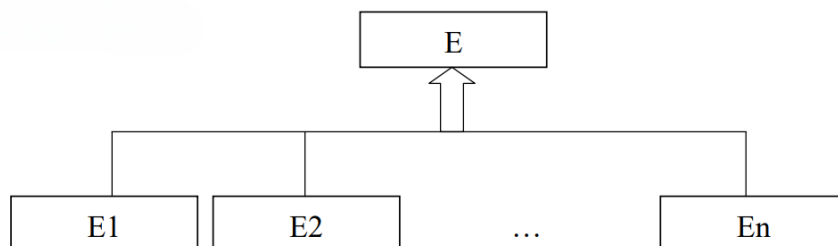
Esempio di cardinalità in un'associazione.

Nel caso di un'associazione binaria A tra due entità $E1$ ed $E2$ si dice che A è di tipo:

- **Uno a uno** se le cardinalità massime di entrambe le istanze rispetto ad A sono 1.
- **Uno a molti** se $\text{max-card}(E1, A) = 1$ e $\text{max-card}(E2, A) = n$ o viceversa.
- **Molti a molti** se $\text{max-card}(E1, A) = n$ e $\text{max-card}(E2, A) = n$.

Generalizzazioni

Un'entità E è una **generalizzazione** di un gruppo di entità $E1, E2, \dots, En$ se ogni istanza di $E1, E2, \dots, En$ è anche un'istanza di E . In tal caso le entità $E1, E2, \dots, En$ sono dette specializzazioni di E .



Generalizzazione.

In una generalizzazione le proprietà di E sono **ereditate** da E1, E2, ..., En: ogni Ei ha gli attributi di E e partecipa alle associazioni definite per E.

In una generalizzazione le entità E1, E2, ..., En vengono dette specializzazioni.

Copertura delle generalizzazioni

Per ogni generalizzazione è possibile indicare il proprio tipo di copertura, identificabile tramite due caratteristiche indipendenti.

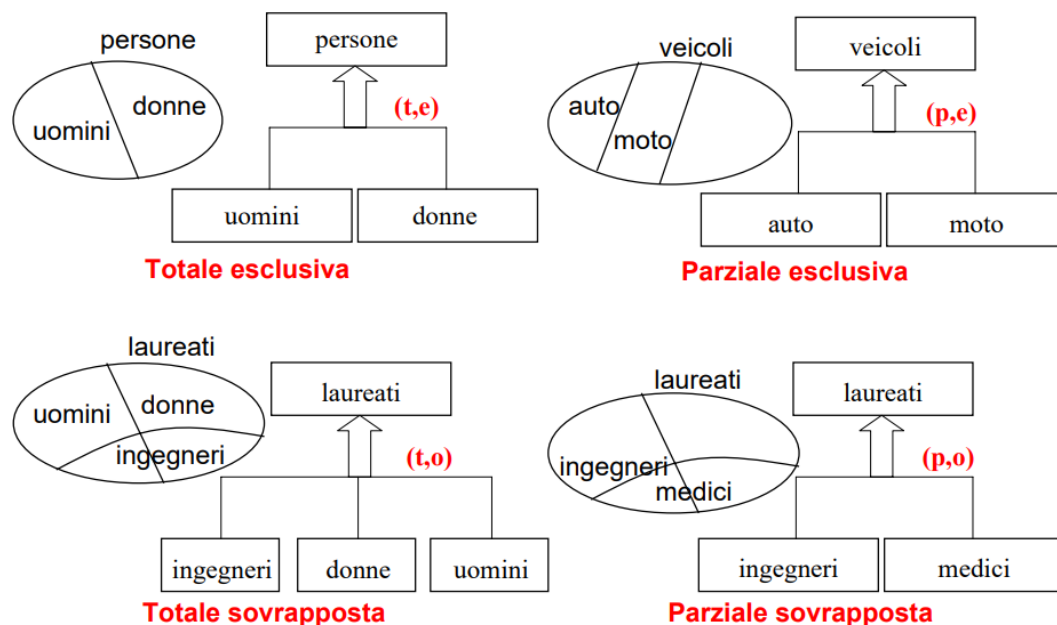
Confronto tra unione delle specializzazioni e generalizzazione:

- **Totale** se la classe generalizzata è l'unione delle specializzazioni.
- **Parziale** se la classe generalizzata contiene l'unione delle specializzazioni.

Indipendenza tra le classi specializzate:

- **Esclusiva** se le specializzazioni sono tra loro indipendenti.
- **Sovrapposta** se esiste un'intersezione non vuota tra le specializzazioni.

Sono dunque possibili le 4 combinazioni (totale, esclusiva), (totale, sovrapposta), (parziale, esclusiva) e (parziale, sovrapposta).



Tipologie di copertura delle specializzazioni.