DATABASE NOTES

Dato: Rappresentazione originaria non interpretata di un evento o di un fenomeno, il fatto che il dato non sia una rappresentazione interpretata implica che bisogna dare una chiave di lettura per poterlo comprendere.

Il software che gestisce una base di dati si chiama *Database Managment System (DBMS)* esso permette di :

- Creare una base di dati
- Manipolare e interrogare i dati presenti
- Condividere i dati garantendo affidabilità, efficienza e privatezza

Il DBMS per garantire l'affidabilità dei dati fa dei controlli di ridondanza (essi hanno come effetti collaterali : spazio sprecato e inconsistenza dei dati); esiste inoltre la ridondanza controllata : ossia si inseriscono dati duplicati appositamente.

Modello concettuale: Ossia il modello ER (Entità Relazione) esso è stato ideato da P.P Chen nel 1976, esso possiede i seguenti costrutti :

- Entità (concetto o oggetto nel mondo reale)
- Associazione (tra due o più entità)
- Attributo (proprietà di una entità o associazione)

Il modello concettuale porta un paio di vantaggi sostanziali : esso migliora la comunicazione tra progettista e utente finale (il grafico è concettuale, quindi anche i meno esperti possono interpretarlo), e inoltre avere una rappresentazione indipendente dalla scelta della struttura dei dati.

Passi principali per la costruzione di una base di dati :

- Analisi dei requisiti : In questa fase si raccolgono e analizzano requisiti relativi ai fabbisogni informativi degli utenti.

- Progettazione concettuale : L'obbiettivo è tradurre i requisiti in una descrizione formale
  e indipendente dalle scelte relative alla realizzazione fisica.

- Progettazione logica : L'obbiettivo è di tradurre lo schema concettuale in una struttura logica per poi poter implementare nella prossima fase il DBMS.

- Progettazione fisica: Implementazione dello schema logico sul determinato DBMS

  Strategie di progetto:

  Top-down: Schema concettuale prodotto da una serie di raffinamenti a partire da uno schema che descrive concetti molto astratti. Utile quando si possiede sin dall'inizio una visione globale di tutte le componenti. Da un livello all'altro vengono eseguite delle primitive elementari:

  • Definizione degli attributi di un'entità o relazione.
  • Reificazione di un attributo o entità.
  • Decomposizione di una relazione in due distinte.
  • Trasformazione di un'entità in una gerarchia di generalizzazione

  Bottom-up: Le specifiche iniziali sono divise in componenti via via sempre più piccole finché non diventano elementari. Vengono rappresentati da semplici schemi concettuali che poi vengono fusi nello schema finale. Primitive di trasformazione:

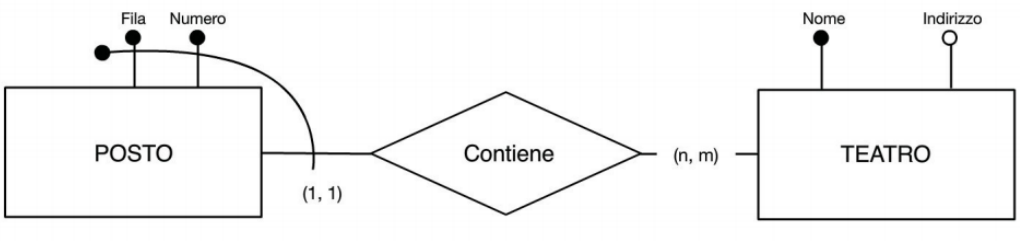
  • Introduzione di una nuova entità o relazione.
  • Individuazione nelle specifiche di un legame tra entità una generalizzazione.
  • Aggregazione di una serie di attributi in una entità o in una relazione.

  Inside-out: Caso particolare di bottom-up, si individuano inizialmente solo alcuni concetti importanti e poi si procede "a macchia d'olio". Cioè si rappresentano prima i concetti in relazione con i concetti iniziali, per poi muoversi verso quelli più lontani attraverso una "navigazione" tra le specifiche.

  Qualità di uno schema concettuale:

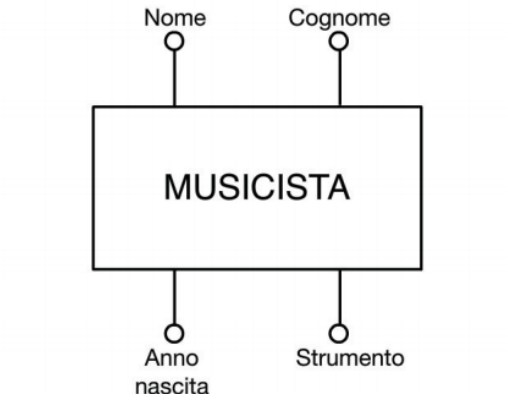
  • Correttezza: deve utilizzare propriamente i costrutti del modello concettuale, possono esserci errori sintattici o semantici.
  • Completezza: deve rappresentare tutti i dati, e tutte le operazioni possono essere eseguite.
  • Leggibilità: scelta opportuna dei nomi, disporre i costrutti in griglia scegliendo quelli centrali quelli con più legami, tracciare solo linee perpendicolari, disporre le entità genitori sopra elle entità figlie.
  • Minimalità: tutte le specifiche sui dati sono rappresentate una volta sola nello schema.

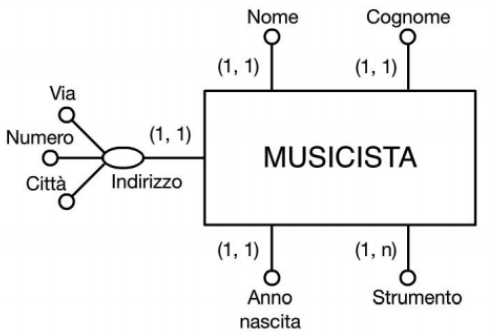
Entità: essa rappresenta una classe di oggetti fisica o concettuale, gli attributi invece sono descritti con i pallini e sono relativi alla entità.
Entità debole: Essa non è identificata solo da se stessa, bensi da una entità forte (attributi entità debole + identificativo entità forte), la sua partecipazione (cardinalità) deve esse (1,1) con l'entità forte.



L'entità debole può essere sempre trasformata in una entità forte aggiungendo un attributo fittizio che fa da identificatore

Attributi : essi possono essere atomici/semplici o composti, i numeri che sono affiancati all'attributo sono la cardinalità minima/massima, ossia quanto quell'attributo può assumere, se la cardinalità non è esplicita essa è (1,1).
Essi possono essere inoltre obbligatori o opzionali (0,x).

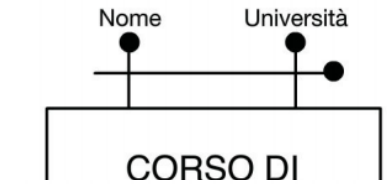




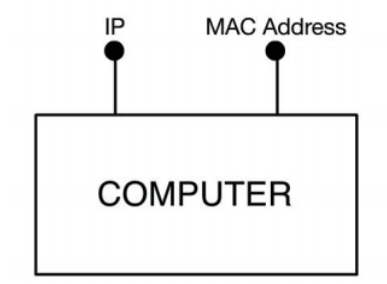
Identificatore di entità: (esso è rappresentato con il pallino nero)

- Per poter identificare una entità dobbiamo trovare un sottoinsieme(minimo) di attributi che identificano gli elementi
- La scelta degli attributi che identificano gli elementi deve essere indipendente dal particolare campione

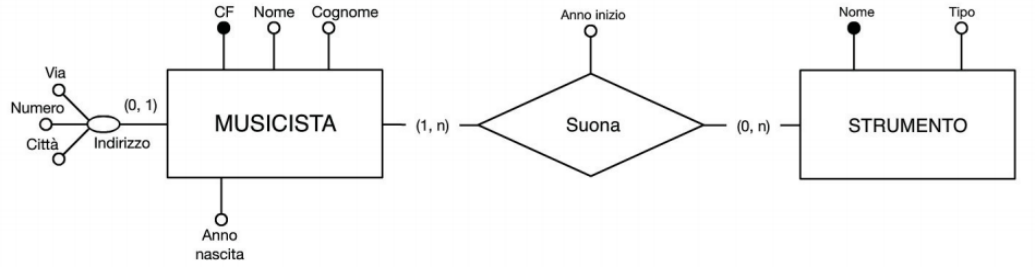
Esso è un attributo composto, ossia il corso di laurea è identificato da Nome e Università insieme.



In questo caso ci sono due identificatori indipendenti.



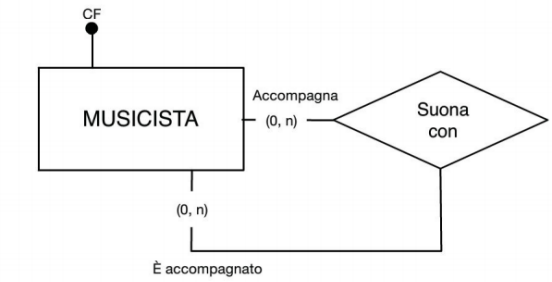
Associazioni : Ci permettono di esplicitare un legame tra due o più entità



Suona è un insieme di elementi dati dal prodotto cartesiano tra gli elementi di musicista e gli elementi di strumento, in questo insieme non è possibile trovare elementi uguali.

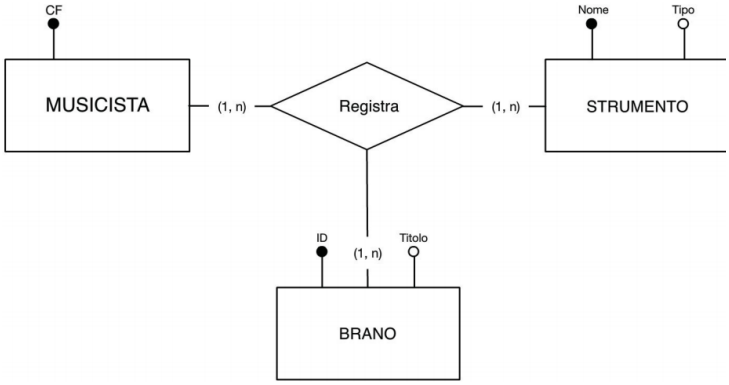
Le cardinalità delle associazioni devono essere scelte con cura, soprattutto la cardinalità minima, essa definisce l'esistenza o meno di elementi dell'entità, inoltre è possibile aggiungere attributi anche direttamente sulla associazione(viene utilizzato quando si ha una proprietà dell'associazione da esplicare).

Le associazioni possono inoltre essere ricorsive :



Le associazioni possono inoltre essere n-arie :

Questo accade quando un'associazione collega più entità.



per questa associazione?

Come scegliere le cardinalità di associazioni n-arie :

- Ci si posiziona su una entità e ci si chiede : dato un elemento di questa entità, quante sono le coppie di elementi delle altre due entità che posso avere

Le associazioni ternarie talvolta sono ostiche da gestire e in alcuni casi è possibile trasformare queste associazioni in associazioni binarie, per esempio quando l'associazione partecipa con cardinalità massima uguale a 1. Per modificare l'associazione bisogna fare delle considerazioni sulle varie cardinalità e creare al posto di una singola ternaria, una coppia di binarie.

Generalizzazioni e specializzazioni

IS-A : Si vuole descrivere una entità che è sotto-insieme di un'altra entità, per questo genere di generalizzazione vale il principio di ereditarietà, ossia ogni istanza dell'entità figlia ha tutti gli attributi dell'entità padre.

Una generalizzazione può essere totale(freccia nera) o parziale, nel caso della generalizzazione totale ogni elemento dell'entità padre deve essere presente in almeno una entità figlia, quindi l'unione dei sotto-insiemi dati dalle entità figlie da l'insieme dell'entità padre.

Nel caso della generalizzazione parziale invece esistono elementi del padre che non sono presenti nelle entità figlie.

Inoltre una generalizzazione può essere disgiunta o sovrapposta, se disgiunta l'intersezione di qualsiasi coppia di sotto-insiemi dell'entità padre è vuota, nel caso sovrapposta esiste almeno un elemento condiviso da più entità figlie.

MODELLO RELAZIONALE

L'idea di questo modello è quella di descrivere un database come un insieme di predicati che descrivono i vincoli sui possibili valori e le possibili combinazioni di un insieme finito di variabili.

Che cosa è una relazione? Essa è un sottoinsieme del prodotto cartesiano di due o più insiemi.

Ogni elemento di una relazione si chiama tupla(n-upla), ossia una sequenza ordinata di valori di attributi (nome,tipo dato), la struttura della relazione si chiama *schema di relazione* R(A1,..,An) dove R è il nome della relazione e A1, ...,An attributi, il # di attributi è il grado della relazione.

La normale rappresentazione di una relazione è una tabella, il nome della relazione è il nome della tabella e gli attributi i nomi delle colonne, le righe invece corrispondono alla tuple dell'istanza.

Se non si ricorda o non si ha ancora un valore da inserire nella tabella di relazione si può sostituire con la parola NULL, esso può avere diversi significati:

- Valore dell'attributo è sconosciuto
- Valore dell'attributo esiste ma non è disponibile
- L'attributo non è applicabile alla tupla

Uno *schema di database* è definito dall'insieme degli schemi di relazione R1(x1),R2(x2) .., Rn(xn)

VINCOLI MODELLO RELAZIONALE

Vincoli intra-relazionali :

- **Vincolo di dominio** : esso specifica il dominio di appartenenza dell'attributo, a questo si possono aggiungere anche ulteriori limitazioni (Es. se il dominio è quello delle stringhe, restringere a 3-5 caratteri)
- **Vincolo di chiave :**
  - Superchiave SK : è un sottoinsieme di attributi della relazione t.c. per ogni coppia di tuple ti,tj con i != j $\sqsupset$ ti[SK]!=tj[SK] , dunque se si aggiunge o toglie un elemento da SK, essa varia.
  - Chiave K : è una superchiave minimale (cioè non è possibile rimuovere nessun elemento senza perdere il vincolo di unicità)

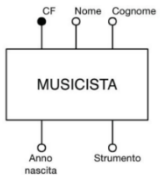
- Sulla chiave primaria inoltre esiste un vincolo che non permette valori nulli, per evitare situazioni di ambiguità

Vincoli inter-relazionali :

- Vincolo di integrità referenziale : esso ha come obbiettivo quello di mantenere coerenti i dati che fanno riferimento ad uno stesso concetto, quando quel concetto è presente in più relazioni, esso si basa sulla chiave esterna (FK) essa è un insieme di attributi che deve soddisfare :
  - La cardinalità di FK = cardinalità di K , e il tipo di attributi deve essere lo stesso
  - T1[FK]=T2[K] (oppure NULL)

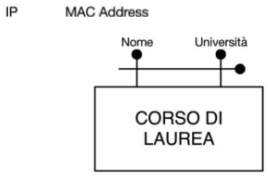
Traduzione ER ⌴ Relazionale

Traduzione entità forte :

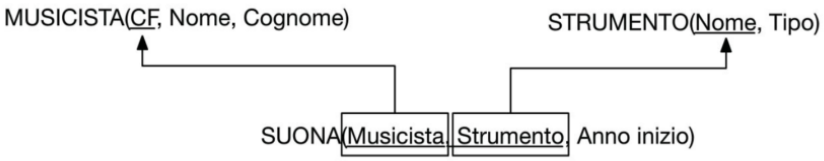


MUSICISTA(CF, Nome, Cognome, Anno nascita, Strumento)

Traduzione associazione (x,n) – (y,n)



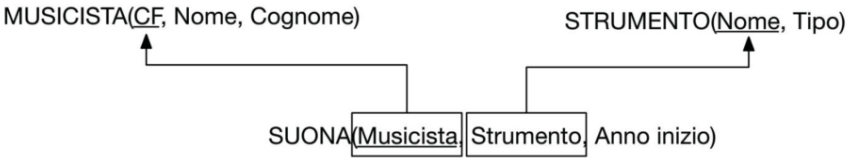
COMP CORSO DI LAUREA(Nome, Università)



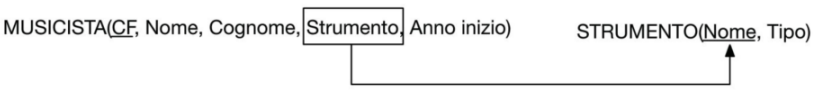
Traduzione associazione binaria (x,1) – (y,m) e (x,1) – (y,1)

Potendo avere più partecipazioni diverse nel caso :

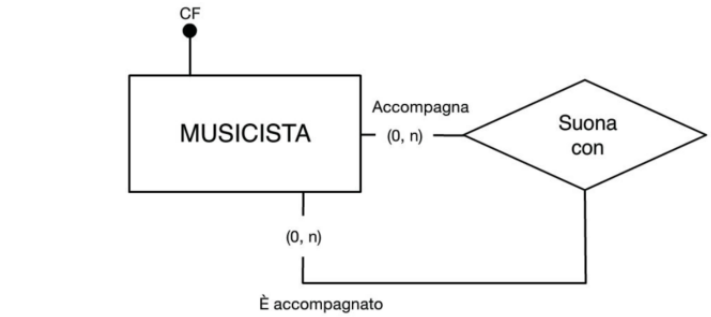
- (1,1) la soluzione ottima è quella con solo due schemi relazionali
- (0,1) ci possono essere due casi da tenere in considerazione :
  - Se la quantità dei valori nulli è piccola rispetto al totale allora optiamo per la soluzione con due schemi altrimenti quella con tre è preferibile

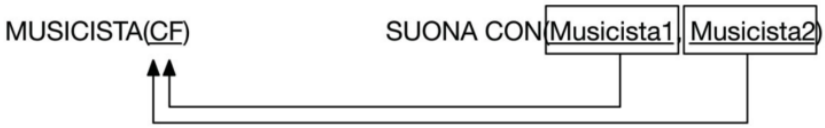


Nel caso (1,1) – (y,m)

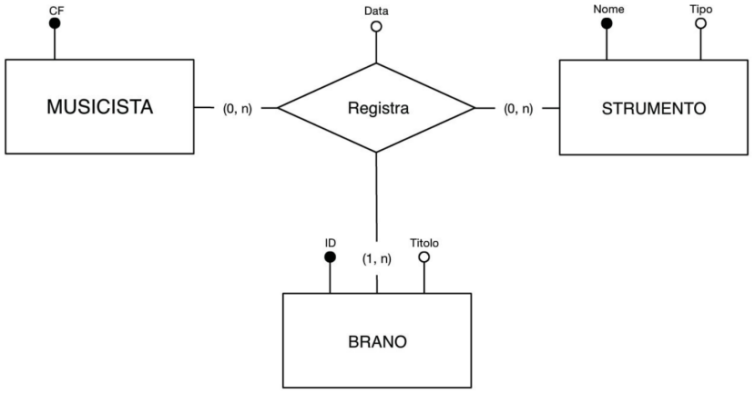


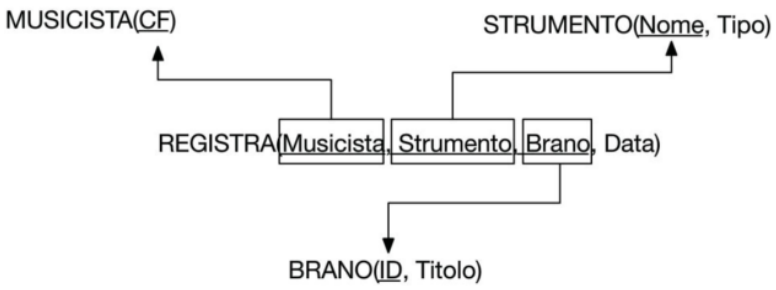
Traduzione associazione ricorsiva (x,n) – (y,m)



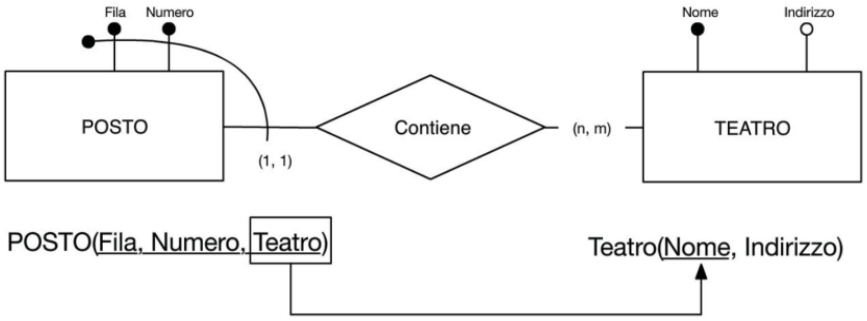


Traduzioni ternarie (x,n) – (y,m) – (z,o)





Traduzione entità debole



ALGEBRA RELAZIONALE

L'algebra relazionale si basa su una logica basata su tre valori, vero, falso e valore nullo. Le operazioni sulle relazioni restituiscono come output altre relazioni.

Le operazioni relazionali sono:

- Operazioni di base : selezione,proiezione, ridenominazione
- Operazioni insiemistiche: unione,differenza,intersezione
- Operazioni di correlazione: prodotto cartesiano, join naturale, theta join
- Funzioni di aggregazione e raggruppamento

Selezione (righe)

$$\sigma_F(r(X))$$

Questa operazione di selezione viene applicata a r(X) e seleziona le righe della tabella che verificano la funzione F.

Nota bene:

- La cardinalità $|\sigma_F(r(X))| \leq |r(X)|$
- Il grado è il grado di R(x)

Proiezione (colonne)

$$\pi_Y(r(X))$$

Questa operazione di proiezione viene applicata a r(X) e proietta tutti le colonne del sottoinsieme di attributi indicati da Y.

Nota bene:

- Con questa operazione se l'insieme Y proiettato ha tuple con nomi duplicati essi collassano in un'unica tupla, perché non possono esistere tuple duplicate.
- La cardinalità di questa operazione è uguale a quella di r(X) se l'insieme Y è superchiave di r(X).

Ridenominazione

$$\mathrm{MUS} \leftarrow \sigma_{Anno\ nascita \leq 1980}(\mathrm{MUSICISTA})$$

La ridenominazione è una operazione unaria.

Operazioni insiemistiche

Esse possono essere applicate solo su insiemi che hanno lo stesso grado(stesso numero di attributi) e che gli attributi siano dello stesso tipo.

Quando si applicano queste operazioni è cura rinominare gli attributi che hanno nomi diversi ma hanno dati con lo stesso significato (Es. data di nascita e anno di nascita).

Le tuple duplicate vengono collassate.

- l'unione di due relazioni $r_1(X)$ e $r_2(X)$ definite sullo stesso insieme di attributi X è

$$r_1(X) \cup r_2(X) = \{t \mid t \in r_1(X) \text{ OR } t \in r_2(X)\}$$

- la differenza tra due relazioni $r_1(X)$ e $r_2(X)$ è

$$r_1(X) \setminus r_2(X) = \{t \mid t \in r_1(X) \text{ AND } t \notin r_2(X)\}$$

- l'interseziona tra due relazioni $r_1(X)$ e $r_2(X)$ è

$$r_1(X) \cap r_2(X) = \{t \mid t \in r_1(X) \text{ AND } t \in r_2(X)\}.$$

Operazioni di correlazione

Join

Questa operazione permette di mettere insieme i dati di più tabelle.

- Prodotto cartesiano : è una operazione binaria e combina ogni elemento di una relazione con tutti gli elementi dell'altra relazione(i due insiemi devono essere disgiunti)
  - La relazione risultante avrà grado pari alla somma dei gradi
  - Cardinalità uguale al prodotto delle cardinalità

$$\text{MUSICISTA} \times \rho_{(Nom,Cog,DataNascita)}(\text{CANTANTE})$$

- Join naturale : questa operazione correla i dati di due tabelle in base a valori uguali di attributi che hanno lo stesso nome e sono compatibili
  - Lo schema prodotto ha come grado un valore pari alla cardinalità dell'unione degli insiemi.
  - La cardinalità:
    - Se il join è fatto su attributi che non sono superchiave è possibile che non ci siano valori uguali $\sqcup$ relazione vuota
    - Se il join viene eseguito sulla superchiave -> $|r_1(X) \bowtie r_2(Y)| \leq |r_2(Y)|$
    - Se l'intersezione degli insiemi è chiave primaria allora ->

$$|r_1(X) \bowtie r_2(Y)| = |r_2(Y)|$$

$$r_1(X) \bowtie r_2(Y) = \{t \mid t \in r(X \cup Y),\ t[X] \in r_1(X) \text{ AND } t[Y] \in r_2(Y)\}$$

- Theta-join: Esso è un join condizionato, in questo modo quando le tabelle vengono intersecate esso elimina in fase di costruzione tutte le tuple che non soddisfano la condizione

$$\text{MUSICISTA} \bowtie_{CF=Musicista} \text{CANTANTE}$$

- Join incompleto o esterno: Questo tipo di join è incompleto in quanto risultano tuple pendenti, ossia attributi non chiave nulli, esso può essere di tre tipi:
  - Join sinistro: Si tengono tutte le tuple della tabella a sinistra
  - Join destro: Si tengono tutte le tuple della tabella a destra
  - Join completo: Si tengono tutte le tuple pendenti

Operazioni di aggregazione e raggruppamento

- Proiezione generalizzata: $\pi_{Data,Teatro,Prezzo*Biglietti}(\text{MUSICISTA})$

  Essa oltre a proiettare data e teatro, calcola prezzo*biglietti e lo proietta.

- Operazioni di aggregazione:

$$\mathcal{F}_{SUM(Ricavo),AVERAGE(Ricavo)}(\rho_{(Data,Teatro,Ricavo)}(\text{MUS}))$$

  - SUM
  - MAX
  - MIN
  - AVERAGE
  - COUNT

- Operazioni di raggruppamento :

$$_{(Teatro)}\mathcal{F}_{SUM(Ricavo),COUNT(Teatro)}(\text{INCASSO})$$

  Aggrego i ricavi e conto i teatri, infine raggruppo per teatro. Dunque per ogni teatro conto quanti eventi ci sono stati (COUNT(teatro)) e sommo(Ricavi).

  Le relazioni calcolate avranno :

  - Grado pari alla somma della lista degli attributi di raggruppamento

- La cardinalità sarà il numero di valori distinti sugli attributi di raggruppamento

Algebra relazionale e valori nulli

I valori nulli possono:

- Rappresentare un valore sconosciuto (esiste ma non sappiamo il valore)
- Può non essere disponibile
- Può essere un valore non applicabile

Comportamento selezione: Scarta le tuple che restituiscono NULL, se vogliamo includere i valori sconosciuti si aggiunge IS NOT NULL/IS NULL.

Comportamento proiezione: Elimina le tuple duplicate, ma mostra comunque i valori nulli.

Comportamento operazioni insiemistiche/join: Si comportano secondo la logica dei tre valori.

Comportamento aggregazione/raggruppamento:

- SUM,MAX,MIN,AVERAGE si comportano come se il valore null non esistesse
- COUNT si comporta "normalmente" se conto su tutti i valori o su una superchiave, altrimenti (se non superchiave) essa scarta le tuple con il NULL.

Logica a tre valori: In questa logica il valore sconosciuto è trattato come falso, infatti esiste la relazione IS NULL o IS NOT NULL

AND	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

OR	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

NOT	
T	F
F	T
U	U

SQL: esso è un linguaggio strutturato di interrogazione

Il DBMS è un software che permette la gestione di una base di dati esso fornisce più linguaggi per implementare la base di dati :

- DDL (Data definition language) : Usato per specificare lo schema concettuale, che consente la descrizione delle entità,associazioni e vincoli.
- SDL (Storage definition language): Usato per specificare schemi interni
- VDL (View definition language): Usato per specificare schemi esterni

- DML (Data manipulation language): Usato per manipolare i dati, inserimento/cancellazione ecc..

Definizione dati in SQL: (case sensitive)

- CREATE DATABASE
- DROP DATABASE
- CREATE TABLE
- DROP TABLE
- ALTER TABLE : modifica lo schema della tabella
-

```
CREATE TABLE Musicista (
  CF     CHAR(5),
  Nome   VARCHAR(30),
  Cognome VARCHAR(30)
);
```

```
ALTER TABLE Musicista ADD DataNascita DATE;
```

INSERT

```
INSERT INTO <nome tabella>(<elenco attributi>) VALUES
  (<elenco valori da inserire negli attributi>);
```

I tipi di dati supportati sono : smallint,boolean,numeric,float,double,real,precision numeric,integer,bigint,character,varchar,date e time alcuni DBMS inoltre supportano anche +/- infinity e NaN.

In SQL si possono inserire vincoli di dato non nullo inserendo NOT NULL nella lista di attributi desiderata nella create table.

```
CREATE TABLE Musicista (
  CF     CHAR(5),
  Nome   VARCHAR(30) NOT NULL,
  Cognome VARCHAR(30) NOT NULL
);
```

Si possono inoltre inserire altri vincoli, ad esempio limitare l'inserimento a alcune stringhe specifiche, esso viene fatto tramite la clausola CHECK :

```
CREATE TABLE Musicista (
  CF     CHAR(5),
  Nome   VARCHAR(30) NOT NULL,
  Cognome VARCHAR(30) NOT NULL,
  Sesso CHAR(1) CHECK (Sesso='M' OR Sesso='F')
);
```

Vincolo di chiave primaria :

```
CREATE TABLE Suona (
  Musicista CHAR(5) NOT NULL,
  Strumento VARCHAR(30) NOT NULL,
  AnnoInizio SMALLINT CHECK (AnnoInizio>=1900),
  PRIMARY KEY (Musicista,Strumento)
);
```

Vincolo UNIQUE esso permette di inserire chiavi candidate che non sono primarie :

```
CREATE TABLE Musicista (
  CF     CHAR(5) PRIMARY KEY,
  Nome   VARCHAR(30) NOT NULL,
  Cognome VARCHAR(30) NOT NULL,
  NumeroTessera INTEGER UNIQUE NOT NULL CHECK (NumeroTessera
    >0)
);
```

Vincoli integrità referenziale:

```
CREATE TABLE Suona (
  Musicista CHAR(5) NOT NULL,
  Strumento VARCHAR(30) NOT NULL,
  AnnoInizio SMALLINT CHECK (AnnoInizio>=1900),
  PRIMARY KEY (Musicista,Strumento),
  FOREIGN KEY (Musicista) REFERENCES Musicista(CF)
    ON DELETE NO ACTION ON UPDATE CASCADE,
  FOREIGN KEY (Strumento) REFERENCES Strumento(Nome)
    ON DELETE NO ACTION ON UPDATE CASCADE
);
```

Opzioni :

- NO ACTION : rifiuta l'operazione(comportamento di default)
- CASCADE : richiede cancellazione/aggiornamento delle righe che hanno chiave esterna collegata alle righe modificate nella tabella primaria.
- SET NULL : il valore delle righe coinvolte viene settato a NULL

MANIPOLAZIONE DATI IN SQL

```
SELECT <lista di attributi>
FROM <lista di tabelle>
WHERE <condizione di selezione>
GROUP BY <lista di attributi (di raggruppamento)>
HAVING <condizione (di raggruppamento)>
ORDER BY <lista di attributi>
```

- Selezione si comporta esattamente come quella del modello relazionale, si può condizionare in base alle preferenze con la clusola WHERE.
- Proiezione invece non elimina le tuple uguali quindi se la richiesta chiede solo tuple diverse una dall'altra, la clausola DISTINCT fa al caso nostro e viene applicata in questo modo:

```
SELECT DISTINCT Nome, Cognome, AnnoNascita
FROM Musicista;
```

- Ridenominazione :

```
SELECT CF as "CodFis", Cognome AS "Cog"
FROM Musicista;
```

- Unione/Except/Intersect :

  I duplicati vengono eliminati in automatico a meno che non si usi la clausola ALL dopo UNION.

```
SELECT Nome, Cognome, AnnoNascita
FROM Musicista
UNION
SELECT Nome, Cognome, DataNascita AS AnnoNascita
FROM Cantante;
```

- Join:

```
SELECT *
FROM Musicista NATURAL JOIN Suona;
```

- Theta Join

```
SELECT *
FROM Musicista
  JOIN Suona ON Musicista.CF=Suona.Musicista;
```

- Outer Join (se a sinistra LETF OUTER JOIN, altrimeti RIGHT OUTER JOIN

```
SELECT *
FROM Musicista
  FULL OUTER JOIN Strumento ON Musicista.Strumento=Strumento.
    Nome;
```

- Aggregazione

```
SELECT Data, Teatro, Prezzo * BigliettiVenduti
FROM Concerto;
```

- SUM(<expressione>) per sommare i valori delle righe su un attributo di una tabella o ottenuti come risultato di un'espressione; l'espressione può coinvolgere anche più attributi come nel caso del prodotto tra 'Prezzo' e 'Biglietti Venduti';
- MAX(<expressione>) per trovare il massimo tra i valori dell'attributo specificato o tra i valori ottenuti come risultato di un'espressione;
- MIN(<expressione>) per trovate il minimo tra i valori dell'attributo specificato o tra i valori ottenuti come risultato di un'espressione;
- AVG(<expressione>) per calcolare la media dei valori dell'attributo specificato o dei valori ottenuti come risultato di un'espressione;
- COUNT(*) per contare quante righe ci sono in una relazione;
- COUNT(DISTINCT <nome attributo>) per contare il numero di valori distinti assunti dall'attributo a cui viene applicata la funzione;
- COUNT(<nome attributo>) per contare il numero di valori non nulli assunti dall'attributo a cui viene applicata la funzione (i duplicati non verranno eliminati).

- Raggruppamento (la clausula GROUP BY raggruppa per Teatro in questo specifico caso)

```
SELECT Teatro, SUM(Prezzo*BigliettiVenduti) AS Totale, COUNT
    (*) AS Concerti
FROM Concerto
GROUP BY Teatro;
```

Anche il GROUP BY può essere condizionato, utilizzando la clausola HAVING :

```
SELECT Teatro, SUM(Prezzo*BigliettiVenduti) AS Totale, COUNT
    (*) AS Concerti
FROM Concerto
GROUP BY Teatro
HAVING COUNT(*)>1;
```

Nota bene : prima viene valutata la clausola WHERE e quindi viene fatto il raggruppamento prima condizionato da WHERE e poi dalla clausola HAVING.

- Ordinamento(NaN maggiore di ogni altro valore)

```
SELECT *
FROM Concerto
ORDER BY BigliettiVenduti DESC, Teatro ASC;
```

- Pattern Matching

```
SELECT *
FROM Musicista
WHERE Nome LIKE 'Al%';
```

% rimpiazza zero o più caratteri

_ rimpiazza un solo carattere

Normalizzazione

Le forme normali certificano la qualità di uno schema relazionale, quando una relazione non soddisfa una forma normale, allora presenta delle ridondanze, ci sono inoltre vari tipi di anomalie :

- Ridondanza
- Anomalia di aggiornamento
- Anomalia di cancellazione
- Anomalia di inserimento

Dipendenza funzionale : Y $\sqsupset$ Z è un particolare vincolo di integrità che descrive i legami di tipo funzionale tra gli attributi di una relazione. Data una relazione r su uno schema R(X) e due sotto insiemi non vuoti Y e Z di X, esiste su r una dipendenza funzionale tra Y e Z se per ogni coppia di tuple t1 e t2 di r aventi gli stessi valori sugli attributi di Y, risulta che hanno gli stessi valori anche su Z,

Una dipendenza Y $\sqsupset$ A non è banale se A compare tra gli attributi di Y

Decomposizione:

Data una relazione r su un insieme di attributi X, $se\ X_1 \cup X_2 = X\ con\ X_1, X_2 \subseteq X$, allora il join delle due relazioni ottenute per proiezione da r su X_1 e X_2 è una relazione che contiene tutte le tuple di r, più eventualmente altre dette "spurie".

r si **decompone** senza perdita su X_1 e X_2: se il join delle due proiezioni è uguale ad r stessa, cioè non contiene spurie. Ovvero quando: $X_0\ soddisfa\ X_0 \rightarrow X_1\ oppure\ X_0 \rightarrow X_2\ con\ X_0 = X_1 \cap X_2$.

Quindi, quando decomponiamo una relazione in due parti, se l'insieme degli attributi comuni è chiave per almeno una delle due relazioni, allora siamo certi che tutte le istanze della relazione si decompongono senza perdita.

Garantisce che le informazioni nella relazione originaria siano ricostruibili con precisione.

Forma normale di Boyce e Codd: Una relazione r è in forma di Boyce e Codd se per ogni dipendenza funzionale (non banale) X→A definita su di essa, X contiene una

chiave K di r, cioè X è superchiave per r. Anomalie e ridondanze non si presentano in questo tipo di forma normale.

Terza forma normale: Esistono schemi che violano la forma di Boyce e Codd per i quali non esiste alcuna decomposizione che conservi le dipendenze. Quindi si ricorre ad una forma meno restrittiva, che però ha il vantaggio di essere sempre ottenibile. r è in terza forma normale se, per ogni sua dipendenza funzionale X→A, almeno una delle seguenti condizioni è vera:

• X contiene una chiave K di r

• A appartiene ad almeno una chiave di r Se una relazione ha solo una chiave, allora le due forme normali coincidono.

Altre forme normali:

• Prima forma normale: gli attributi delle relazioni sono definiti su valori atomici e non su valori complessi come insiemi o relazioni. Ovvero:

  o Tutte le righe della tabella contengono lo stesso numero di colonne; o Gli attributi rappresentano informazioni elementari;

  o I valori che compaiono in una colonna sono dello stesso tipo, cioè appartengono allo stesso dominio;

  o Ogni riga è diversa da tutte le altre, cioè non ce ne possono essere due con gli stessi valori nelle colonne;

  o L'ordine con il quale le righe compaiono nella tabella è irrilevante

La prima forma normale è già parte integrante della definizione formale di relazione nel modello relazionale.

• Seconda forma normale: variante debole della terza. Una relazione è in seconda forma se su di essa non sono definite dipendenze parziali, cioè dipendenze fra un sottoinsieme proprio della chiave e altri attributi.

Ovvero quando è in prima forma normale e tutti i suoi attributi non-chiave dipendono dall'intera chiave, cioè non possiede attributi che dipendono soltanto da una parte della chiave. La seconda forma normale elimina la dipendenza parziale degli attributi dalla chiave e riguarda il caso di relazioni con chiavi composte, cioè formate da più attributi.

Terza forma normale: Esistono schemi che violano la forma di Boyce e Codd per i quali non esiste alcuna decomposizione che conservi le dipendenze. Quindi si ricorre ad una forma meno restrittiva, che però ha il vantaggio di essere sempre ottenibile.

r è in terza forma normale se, per ogni sua dipendenza funzionale X→A, almeno una delle seguenti condizioni è vera:

- X contiene una chiave K di r
- A appartiene ad almeno una chiave di r

Se una relazione ha solo una chiave, allora le due forme normali coincidono

Tavola degli accessi e tabella dei volumi:

La tavola degli accessi descrive in modo astratto il costo di utilizzo dello schema da parte della operazione, inteso come tempo speso dalla operazione. Il tempo speso (di istanza) è il tempo impiegato per trasferire dati dalla memoria secondaria a quella principale.

Il numero di accessi è dato da accessi*frequenza di accesso.