BASI DI DATI

Prof. R. Orsini

AA 2002/2003

INTRODUZIONE E TERMINOLOGIA

→ sistema informativo di un'organizzazione: combinazione di risorse, umane e materiali, e di procedure organizzate per la raccolta, l'archiviazione, l'elaborazione e lo scambio delle informazioni necessarie alle attività operative, di gestione, di programmazione, di controllo e valutazione dell'organizzazione.

→ sistema informatico: insieme degli strumenti informatici impiegati per il trattamento automatico delle informazioni di un'organizzazione. Sono previste 4 componenti principali:

- *software e hardware* di base
- *base informativa*, contenente la rappresentazione delle informazioni memorizzate
- *schema*, che descrive la struttura della base informativa
- *programmi applicativi*, forniscono i servizi agli utenti eseguendo un certo insieme di operazioni sulla base informativa

→ base di dati: raccolta di dati permanenti suddivisi in:

- *dati*: rappresentazione di certi fatti conformi allo schema
  - sono organizzati in insiemi omogenei, fra i quali sono definite delle relazioni
  - sono molti e non possono essere gestiti tutti insieme nella memoria temporanea
  - sono permanenti finché non vengono esplicitamente rimossi
  - sono accessibili mediante transazioni
  - sono protetti sia da accessi non autorizzati, che da malfunzionamenti hardware e software
  - sono utilizzabili contemporaneamente da utenti diversi
- *metadati*: schema della base di dati, raccolta di definizioni che descrivono la struttura dei dati, le restrizioni sui valori ammissibili dei dati (vincoli d'integrità), le relazioni esistenti fra gli insiemi

Studenti			
Nome	***Matricola***	***Città***	***AnnoNascita***
Isaia	71523	Pisa	1970
Rossi	72345	Lucca	1975
Bianchi	73456	Livorno	1973
Bonin	72457	Pisa	1973

ProveEsami			
Materia	***Matricola***	***Data***	***Voto***
Basi di Dati	71523	12/01/02	22
Basi di Dati	73456	12/01/02	24
Basi di Dati	72345	12/01/02	19
Calcolo I	73456	04/02/02	18
Calcolo I	72457	04/02/02	28

I ***dati*** sono rappresentati dalle righe delle tabelle, contenenti le informazioni relative ai singoli studenti ed esami.
I ***metadati*** riguardano, invece, le seguenti informazioni:

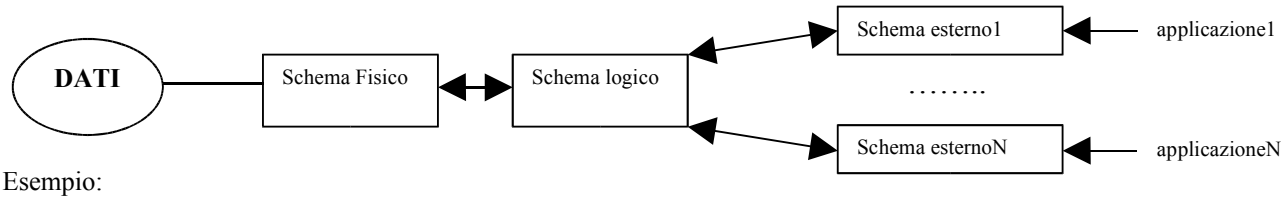
- l'esistenza di due collezioni d'interesse: Studenti e ProveEsami
- la struttura degli elementi di queste due collezioni (intestazioni delle tabelle)
- il fatto che ad ogni esame corrisponde uno studente con la matricola specificata e ad ogni studente corrispondono uno, nessuno o più esami
- l'esistenza di vincoli, quali il fatto che il valore della matricola identifica una sola riga nella tabella Studenti e il valore di matricola e materia identifica una sola riga nella tabella ProveEsami

→ DBMS: sistema centralizzato o distribuito che consente

- di definire schemi di basi di dati
- di scegliere strutture dati per la memorizzazione e l'accesso ai dati
- di memorizzare, recuperare e modificare i dati

3 livelli di descrizione dei dati:

- fisico: viene descritto il modo in cui i dati vanno organizzati nelle memorie permanenti
- logico: viene decritta la struttura e le relazioni che intercorrono tra i dati
- di vista logico (o schema esterno): viene definito come deve apparire la struttura della base di dati ad una certa applicazione



Esempio:

Schema logico:

```
CREATE TABLE Personale
(
        Nome CHAR(30),
        CodiceFiscale CHAR(15),
        Stipendio INTEGER,
        Parametro CHAR(6),
        Recapito CHAR(8)
)
```

Livello di vista logico:

```
CREATE VIEW PersonalePerUfficioStipendi AS
        SELECT Nome, CodiceFiscale, Stipendio, Parametro
        FROM Personale

CREATE VIEW PersonalePerLaBiblioteca AS
        SELECT Nome, Recapito
        FROM Personale
```

Una VIEW è una tabella calcolata da altre con opportuni operatori

→ indipendenza fisica: non è necessario modificare i programmi applicativi quando si modifica l'organizzazione di dati
→ indipendenza logica: non è necessario modificare i programmi applicativi quando si modifica lo schema logico

PROPRIETA' DBMS:

1. *integrità*: garantire che la base di dati si trovi sempre in uno stato il più possibile consistente. Non è necessario solo definire la struttura dei dati, ma anche garantire le condizioni che li rendano significativi.
2. *affidabilità*: garantire l'efficienza della base di dati indipendente da malfunzionamenti hardware e software o da accessi sincroni di più utenti. Le interazioni devono avvenire tramite transazioni (sequenza di azioni in lettura e scrittura delle base di dati e di elaborazione in memoria temporanea):
    a. *atomicità*: annullamento effetti parziali, dovuti a transazioni che vengono terminate prematuramente
    b. *serializzabilità*: l'effetto sulla base di dati dell'esecuzione simultanea di più transazioni, è equivalente all'esecuzione seriale di più operazioni
    c. *persistenza*: le modifiche alla base di dati dovute ad una transazione terminata correttamente sono permanenti, indipendenti da malfunzionamenti (fallimenti di transizione (terminazione prematura della transizione), di sistema (hardware/software), disastri (danneggiamento memoria permanente)) successivi alla terminazione
3. *sicurezza*: controllo degli utenti in base al quale può accedere alla base di dati solo l'utente autorizzato e restrizione sui dati accessibili, ovvero sulle operazioni che si possono fare su di essi

MODELLO DEI DATI:

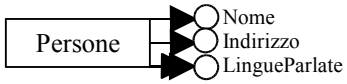
- *entità*: *qualcosa di concreto o astratto* di cui interessa rappresentare alcuni fatti (oggetti concreti, astratti ed eventi)
- *proprietà*: descrive una *qualità* di un'entità. Il valore della proprietà non interessa in quanto tale, ma come caratterizzazione di un'entità
- *tipo*: *descrizione astratta* di ciò che accomuna un insieme di entità omogenee, esistenti o possibili. Un tipo può essere visto come una collezione infinità di entità possibili. Ad esempio il tipo Persona, può essere costituito da Maria, Anna, Sandro… ma anche da tutte quelle entità che non ci sono ancora (tutte le persone che devono ancora nascere ma che nasceranno)
- *collezione*: insieme variabile nel tempo di entità omogenee. L'insieme degli elementi di una collezione in un determinato momento è definito *estensione della collezione*

- *istanza di associazione*: fatto che correla due o più entità, stabilendo un legame logico tra di loro ("Lo studente Rossi frequenta il corso di Programmazione" è un'istanza di associazione espressa dal verbo "frequenta" che lega l'entità "Rossi" con l'entità "Programmazione")
- *associazione*: insieme di istanze di associazione tra elementi di C1….Cn, che varia in generale nel tempo. Il prodotto cartesiano delle estensioni di C1…Cn è detto dominio dell'associazione. Ecco le proprietà strutturali dell'associazione:
    - *molteplicità*: numero massimo di elementi Y che possono trovarsi in relazione con elementi X (multivalore, relazione che coinvolge più elementi; univoca, coinvolge un solo elemento)
    - *cardinalità*: descrive la molteplicità dell'associazione e della sua inversa
        - *uno a molti*: (1:N) se l'associazione è multivalore da X ad Y ed univoca da Y a X
        - *molti ad uno*: (N:1) se l'associazione è univoca da X ad Y e multivalore da Y a X
        - *molti a molti*: (M:N) se l'associazione è multivalore sia da X ad Y che da Y ad X
        - *uno ad uno*: (1:1) se l'associazione è univoca sia da X ad Y che da Y ad X
- *struttura della conoscenza concreta*
- *conoscenza concreta*

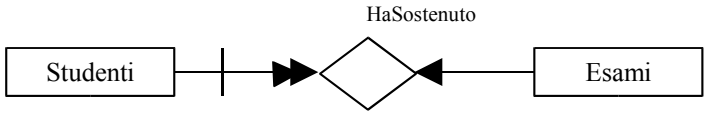
→ Oggetto: rappresentazione nel modello informatico degli aspetti strutturali e comportamentali di un'entità della realtà. E' un'entità software dotato di *stato* (insieme di campi che modellano le proprietà dell'entità), *comportamento* (metodi, le operazioni applicabili all'oggetto vengono chiamate *messaggi*) e *identità* (caratteristica associata all'oggetto fin dalla sua creazione).

→ Interfaccia di un oggetto: insieme dei messaggi a cui esso può rispondere

→ Classe: insieme di oggetti dello stesso tipo, modificabile con operatori per includere o estrarre elementi dall'insieme, al quale sono associati alcuni vincoli d'integrità

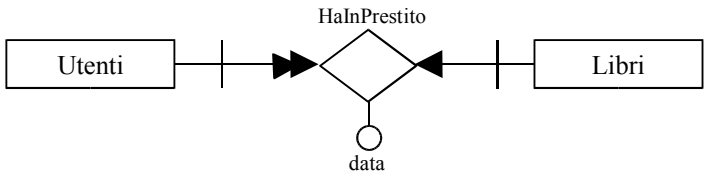


Classe con attributi

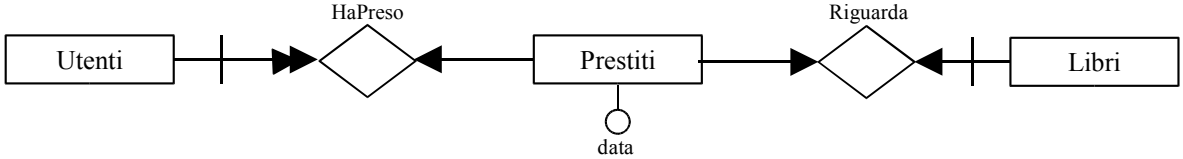


Uno studente ha sostenuto *uno-nessuno-più (1:N)* esami, mentre un esame è stato sostenuto da *uno (1:1)* studente.

→ Associazione binaria: è una relazione binaria fra X ed Y, può non avere proprietà (come nell'esempio precedente) o averle (come nell'esempio successivo).



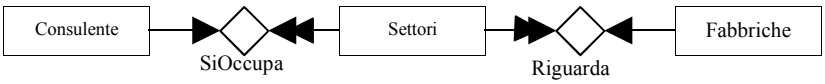
Un utente ha in prestito *uno-nessuno-più* libri (1:N), mentre un libro è prestato ad *uno o nessun* utente (1:1).
Questa associazione è dotata di proprietà, l'attributo *data* relativo al prestito.
Un associazione di questo tipo può essere modellata interpretando l'istanza di associazione "HaInPrestito" come un'entità, definendo così una classe *Prestiti*.



Un utente può avere *uno-nessuno-più* prestiti, mentre un prestito riguarda *un* utente.
Un prestito riguarda *un* libro, mentre un libro è associato ad *uno o nessun* prestito.

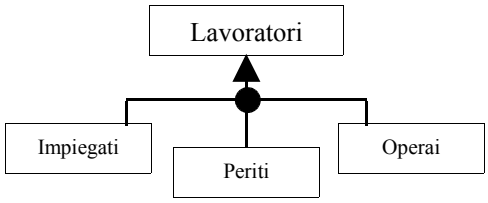
→ Associazione n-aria: è una relazione (N:M) tra X e Y. Può essere generalizzata in questo modo:

Questa è una tipica associazione n-aria (un consulente lavora per una-più fabbriche e una fabbrica ha uno-più consulen ti).



Un consulente si occupa di un settore, mentre un settore è gestito da più consulenti.
Un settore riguarda più fabbriche, mentre una fabbrica è associata ad un settore.

→ SottoClasse e SuperClasse: la sotto classe è un'entità che fa parte della SuperClasse ma ha informazioni/proprietà/attributi ridotti.

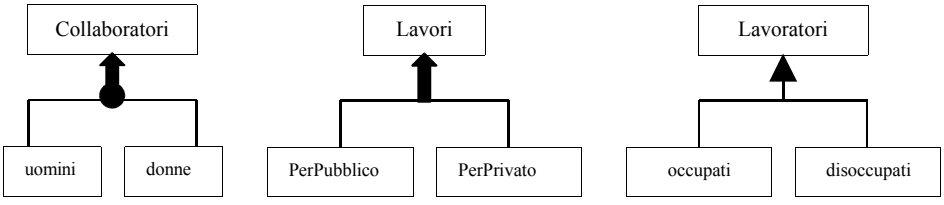


La Super Classe "Lavoratori", può essere suddivisa in 3 sottoClassi: "impiegati", "periti", "operai".

VINCOLI:

1. *vincolo di disgiunzione*: ogni elemento di una sottoClasse di una SuperClasse deve essere disgiunto dagli elementi delle altre sottoClassi della stessa SuperClasse
2. *vincolo di copertura*: gli elementi di tutte le sottoClassi devono coincidere con l'insieme degli elementi della SuperClasse

Quando questi vincoli sono entrambi soddisfatti, si parla di *sottoClassi partizione* della SuperClasse. (1)
Quando viene soddisfatto solo il vincolo di copertura, si parla di *sottoClassi copertura* della SuperClasse. (2)
Quando viene soddisfatto solo il vincolo di disgiunzione, si parla di *sottoClassi disgiunte* della SuperClasse. (esempio sopra)
Quando questi vincoli non sono soddisfatti, si parla di *sottoClassi scorrelate*. (3)



(1) (2) (3)

→ Superchiave: insieme X di attributi tale che i valori degli attributi in X individuano univocamente un'ennupla.

→ Chiave: superchiave minimale, nel senso che se si elimina un attributo, i rimanenti non formano più una superchiave. Un attributo che appartiene ad una chiave, è chiamato attributo primo.

→ Chiave Primaria: fa parte dell'insieme delle chiavi e di solito viene preferita quella con il minor numero di attributi.

→ Chiave Esterna: sono chiavi primarie di relazioni R a cui si riferiscono delle relazioni S. Le ennuple della relazione S vengono associate alle ennuple della relazione R riferita ed assumano il valore della chiave esterna in S che equivale al valore della chiave primaria in R.

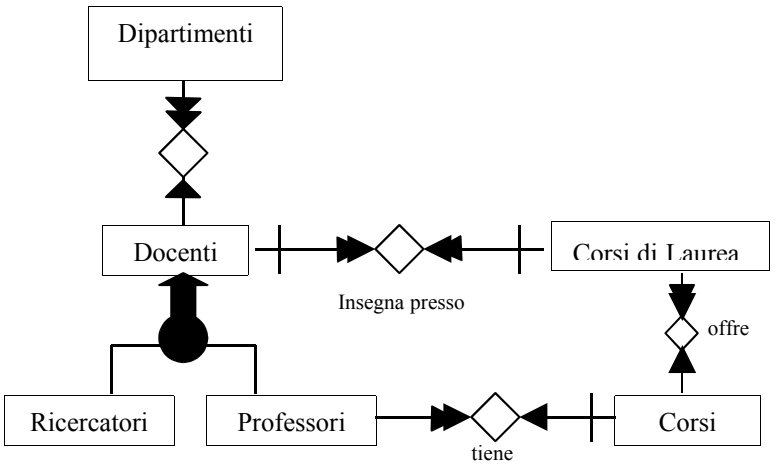
Lavoratori (CodiceFiscale, Nome, NumUfficioUffici*)

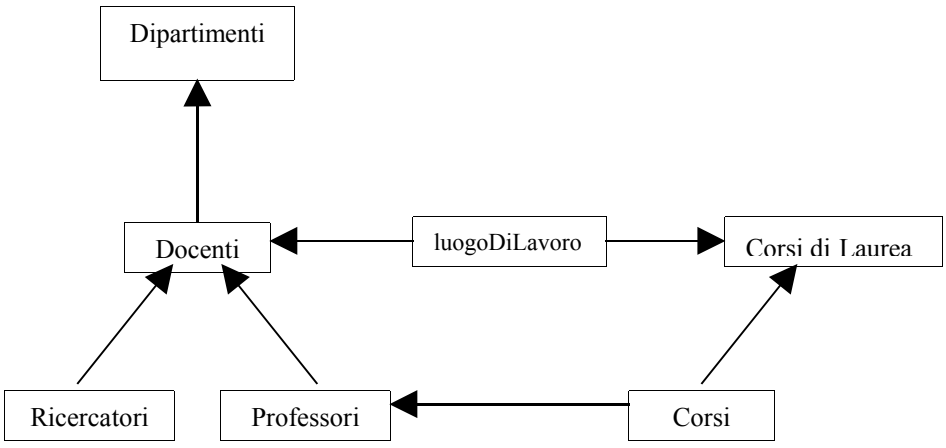
"*NumUfficioUffici*" è chiave esterna per Lavoratori e si riferisce ad un'ipotetica relazione:

Uffici (NumUfficio, NumPersone, SettoreCompetenza)

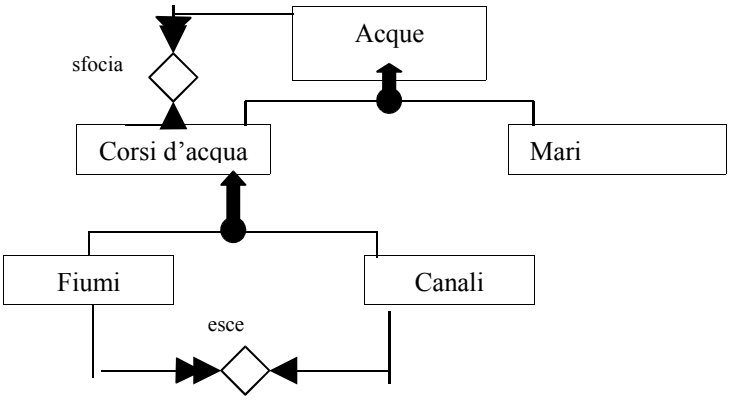
ESEMPI

L'Università è costituita da più dipartimenti a cui corrisponde un nome, un indirizzo e un numero di telefono. Ci sono poi diversi Corsi di Laurea caratterizzati da un nome, un numero di annualità e un numero minimo di esami. Quindi i corsi, con nome e durata. Infine ci sono i docenti che hanno nome, indirizzo e numero di telefono. I docenti si dividono poi in professori (titolari di almento un corso) e ricercatori.





Si vogliono rappresentare informazioni su fiumi e canali. Un fiume è identificato da un nome, ha una lunghezza e la descrizione della sorgente. Un fiume può sfociare nel mare, oppure può immettersi in un altro fiume (affluente). Un canale, identificato da un nome, ha una lunghezza e collega un fiume con un altro fiume oppure un fiume con il mare.



SQL (Structured Query Language)

→ linguaggio universale per la definizione e l'uso di basi di dati relazionali

Una relazione, in SQL, è pensata come una tabella, le cui colonne corrispondono al numero degli attributi delle ennuple (*record*), mentre le righe equivalgono al numero delle ennuple della relazione. Ogni componente di un record è chiamata *campo*.

Clienti (CodiceCliente, CognomeNome, Città, Sconto)
Periti (CodicePerito, CognomeNome, Commissione)
Ordini (NumOrdine, CodicePeritoPeriti*, CodiceClienteClienti*, TipoLavoro, Data, Ammontare)

Utilizzo comando principale: SELECT

```
SELECT *
FROM PERITI
```

Rappresenta una singola tabella, con tutti i record della relazione PERITI (*' * ' include tutti i campi del record*)

```
SELECT *
FROM ORDINI
WHERE Ammontare > 3000
```

Rappresenta una restrizione nella tabella ORDINI che visualizza solo i record che rispettano la condizione (Ammontare>3000)

```
SELECT CognomeNome, Città
FROM Clienti
```

Rappresenta una proiezione della tabella CLIENTI, costituita dai soli campi CognomeNome e Città

```
SELECT *
FROM Clienti, Ordini
```

Rappresenta una tabella che è il prodotto di due tabelle, Clienti e Ordini. Se un perito ha effettuato più ordini verrà ripetuto; per evitare questo basta inserire l'operatore DISTINCT dopo SELECT.

```
SELECT o.NumOrdine, o.Data
FROM Ordini o, Periti p
WHERE p.codicePerito = 'PF2'
        AND o.CodicePeritoPeriti = p.CodicePerito
        AND Ammontare > 3000
```

Restituisce i NumOrdini e la data degli ordini del perito 'PF2' con ammontare maggiore di 3000.

CodicePerito	CognomeNome	Commissione
PF2	Peruzzo Federico	Pubblico
MR3	Mario Rossi	Pubblico
BF2	Bonin Francesco	Privato

NumOrdine	CodicePeritoPeriti	CodiceClienteClienti	TipoLavoro	Data	Ammontare
02342	PF2	Ever34	Interni	12/01/02	4000
03457	BF2	Meb1	Interni	13/01/02	3500
04653	PF2	ComRom	pubblici	13/01/02	8000

Dopo la restrizione...

NumOrdine	Data
02342	12/01/02
04653	13/01/02

AS viene usato per cambiare il nome delle colonne del risultato; nell'esempio precedente:

```
SELECT o.NumOrdine AS Numero Ordine, o.Data AS Data Ordine
FROM Ordini o, Periti p
```

```
WHERE p.codicePerito = 'PF2'
        AND o.CodicePeritoPeriti = p.CodicePerito
        AND Ammontare > 3000
```

Numero Ordine	Data Ordine
02342	12/01/02
04653	13/01/02

IN ci permette di restituire soltanto le espressioni che rispettano la condizione data dalla costante che segue o dalla sottoselect

```
SELECT CodiceCliente
FROM Clienti
WHERE Città IN ('Milano','Pavia','Verona')
```

Mi restituisce i codice dei clienti che si trovano nella città di Milano, Pavia o Verona.

```
SELECT NomeCognome
FROM Clienti
WHERE CodiceCliente IN
        (
        SELECT CodiceCliente
        FROM Ordini
        WHERE Data = '12/01/2002'
        )
```

Mi restituisce il Nome e il Cognome dei Clienti che hanno fatto ordini il 12/01/2002.
Questa SELECT si può anche ottimizzare, senza utilizzare IN, ma il JOIN (o giunzione).

```
SELECT c.NomeCognome
FROM Clienti c, Ordini o
WHERE c.CodiceCliente = o.CodiceCliente
        AND o.Data = '12/01/2002'
```

Il risultato è lo stesso, ma la SELECT risulta ridotta quindi ottimizzata.
Se si volesse ottimizzare ancor più il codice, basterebbe inserire l'operatore DISTINCT subito dopo la SELECT.

```
SELECT DISTINCT c.NomeCognome
FROM Clienti c, Ordini o
WHERE c.CodiceCliente = o.CodiceCliente
AND o.Data = '12/01/2002'
```

NOT EXISTS è un ulteriore operatore che ci permette di restringere i risultati, togliendo tutti quei record che vengono descritti nella sottoselect.

```
SELECT c.CodiceCliente
FROM  Clienti c
WHERE NOT EXISTS
        (
        SELECT *
        FROM Ordini o
        WHERE c.CodiceCliente = o.CodiceCliente
        AND o.Ammontare < 3000
        )
```

Mi mostra tutti i Codici dei Clienti il cui ordine <u>non</u> è inferiore a 3000.

NumOrdine	CodicePeritoPeriti	CodiceClienteClienti	TipoLavoro	Data	Ammontare
02342	PF2	Ever34	Interni	12/01/02	4000
03457	BF2	Meb1	Interni	13/01/02	2500
04653	PF2	ComRom	pubblici	13/01/02	8000

CodiceCliente	CognomeNome	Città	Sconto
Ever34	EverLast	Verona	10%
Meb1	MEB Italia	Verona	5%
ComRom	Comune di Brescia	Brescia	5%

Questo sarà il risultato:

CodiceCliente
Ever34
ComRom

ALL è un operatore di confronto che risulta vero ogniqualvolta la prima espressione sta nella relazione specificata con tutte le espressioni specificate dalla sottoselect.

ANY, invece, risulta vero ogniqualvolta la prima espressione rispetta il criterio di confronto specificato con almeno un'espressione determinata dalla sottoselect. Espr = ANY (Sottoselect) equivale a Espr IN (Sottoselect).
Espr NOT IN (Sottoselect) equivale invece a Espr <> ALL (Sottoselect).

```
SELECT o.NumOrdine, o.Data
FROM Ordini o
WHERE o.Ammontare > ALL
        (
        SELECT o.Ammontare
        FROM Clienti c
        WHERE c.CodiceCliente = o.CodiceCliente
        AND c.Sconto = '10%'
        )
```

Riutilizzando le tabelle di prima, come risultato avremo:

NumOrdine	Data
04653	13/01/02

Infatti solo ComRom (NumOrdine 04653) ha un Ammontare maggiore (8000) rispetto a quello dei clienti con sconto pari al 10% (in questo caso solo Ever34, con Ammontare di 4000).

Per quanto riguarda l'ORDINAMENTO, basta utilizzare l'operatore ORDER BY, con l'attributo DESC per l'ordinamento decrescente.

```
SELECT CodicePerito, CognomeNome
FROM Periti
WHERE Commissione = 'Pubblico'
ORDER BY CognomeNome
```

CodicePerito	CognomeNome
MR3	Mario Rossi
PF2	Peruzzo Federico

```
SELECT CodicePerito, CognomeNome
FROM Periti
WHERE Commissione = 'Pubblico'
ORDER BY CognomeNome DESC
```

CodicePerito	CognomeNome
PF2	Peruzzo Federico
MR3	Mario Rossi

SQL offre inoltre alcune FUNZIONI STATISTICHE predefinite:

- MIN(valore_colonna) → restituisce il minimo valore
- MAX(valore_colonna) → restituisce il massimo valore
- AVG(valore_colonna) → restituisce la media dei valori
- SUM(valore_colonna) → restituisce la somma dei valori
- COUNT(valore_colonna) → restituisce il numero dei valori

```
SELECT MIN(Ammontare), MAX(Ammontare), AVG(Ammontare)
FROM Ordini
WHERE Data = '12/01/02'
```

Restituisce il minimo, il massimo e la media dell'ammontare degli ordini effettuati il 12/01/02.

```
SELECT COUNT( * )
FROM CLIENTI
WHERE Città = 'Verona'
```

Restituisce il numero dei clienti da Verona.

RAGGRUPPAMENTO

GROUP BY va inserito nella SELECT dopo l'eventuale WHERE. Crea delle partizioni che contengono ciascuna le ennuple con lo stesso valore degli attributi di GROUP BY. Viene prodotta un'ennupla per ogni gruppo che soddisfa la condizione di HAVING. Devono essere gli attributi di GROUP BY o una funzione come SUM che restituisce un unico valore per ogni colonna di un gruppo.

```
SELECT CodicePerito, CognomeNome, SUM(Ammontare), AVG(Ammontare)
FROM Ordini
WHERE Data = '12012002'
GROUP BY CodicePerito, CognomeNome
HAVING COUNT( * ) > 5
```

Vengono restituiti Codice del Perito, il Nome e il Cognome, la somma e la media dell'ammontare degli ordini, di tutti i periti che hanno più di 5 ordini in data 12/01/2002.

```
SELECT CodiceCliente, CognomeNome, MAX(Ammontare)
FROM Ordini
WHERE Data IN ('12012002','13012002')
GROUP BY CodiceCliente, CognomeNome
HAVING COUNT( * ) > 2
```

Vengono restituiti Codice del Cliente, il Nome e il Cognome e il massimo dell'ammontare di tutti quelli che anno più di due ordini in data 12/01/2002 o 13/01/2002.

NumOrdine	CodicePeritoPeriti	CodiceClienteClienti	TipoLavoro	Data	Ammontare
02342	PF2	Ever34	Interni	12/01/02	4000
02448	MR3	Meb1	Interni	12/01/02	800
02669	BF2	ComRom	pubblici	12/01/02	1200
03002	PF2	Ever34	Interni	12/01/02	1800
03400	MR3	ComRom	pubblici	13/01/02	2100
03402	PF2	Ever34	Interni	13/01/02	2950
03457	BF2	Meb1	Interni	13/01/02	2500
04653	PF2	ComRom	pubblici	13/01/02	8000
05000	PF2	Meb1	Interni	14/01/02	3000
05220	MR3	ComRom	pubblici	14/01/02	12200

CodiceCliente	CognomeNome	MAX
Ever34	EverLast	4000
ComRom	Comune Brescia	8000

Sia Ever34 che ComRom hanno 3 ordini (quindi HAVING COUNT (*) > 2!!) complessivamente tra il 12/01/02 e il 13/01/02. Le ultime due righe della tabella non vengono considerate.

MODIFICA DATI:

- *Inserzione*
- *Aggiornamento*

- *Cancellazione*

Inserzione:

```
INSERT INTO Clienti
VALUES ('Mir22', 'MiraInfo', 'Venezia', '5%')
```

Aggiornamento :

```
UPDATE Periti
SET CodicePerito = 'PF3'
WHERE CodicePerito = 'PF2'
```

Cancellazione:

```
DELETE
FROM Ordini
WHERE Data < '12012002'
```

CREAZIONE BASI DI DATI

```
CREATE SCHEMA Nome AUTHORIZATION Utente
    Definizioni
```

'Nome' è il nome della Base di Dati
'Utente' è il nome del proprietario della Base di dati
'Definizioni' sono i comandi per la creazione degli elementi della Base di Dati.

RIMOZIONE BASE DI DATI

```
DROP SCHEMA Nome RESTRICT

DROP SCHEMA Nome CASCADE
```

La differenza tra la prima espressione e la seconda sta nel fatto che nella prima, in caso vi siano dati all'interno della base di dati, l'operazione di cancellazione fallisce. Nel secondo caso, la base di dati viene eliminata comunque.

CREAZIONE TABELLA

```
CREATE TABLE Clienti
    (
    CodiceCliente CHAR(3)
    CognomeNome CHAR(30),
    Città CHAR(30),
    Sconto CHAR (4)
    )
```

Per creare una tabella dello stesso tipo di una tabella esistenze, basta utilizzare l'operatore LIKE.

```
CREATE TABLE Ordini2000 LIKE Ordini
AS SELECT *
FROM Ordini
WHERE Data < '01012001' AND Data > '31121999'
```

Come risultato avrò una tabella con gli stessi campi della rabella ORDINI ma avente come dati soltanto quelli relativi all'anno 2000.

ELIMINAZIONE TABELLA

```
DROP TABLE Name
```

CREAZIONE TABELLE VIRTUALI (VIEW)
Le view, viste, non sono delle vere tabelle memorizzate, ma delle costruzioni virtuali che, quando richieste, vengono utilizzate alla stregua di una tabella reale.

```
CREATE VIEW OrdiniPerCliente (CodiceCliente, TotaleOrdini)
AS SELECT CodiceCliente, SUM(Ammontare)
FROM Ordini
GROUP BY CodiceCliente
```

NumOrdine	CodicePeritoPeriti	CodiceClienteClienti	TipoLavoro	Data	Ammontare
02342	PF2	Ever34	Interni	12/01/02	4000
02448	MR3	Meb1	Interni	12/01/02	800
02669	BF2	ComRom	pubblici	12/01/02	1200
03002	PF2	Ever34	Interni	12/01/02	1800
03400	MR3	ComRom	pubblici	13/01/02	2100
03402	PF2	Ever34	Interni	13/01/02	2950
03457	BF2	Meb1	Interni	13/01/02	2500
04653	PF2	ComRom	pubblici	13/01/02	8000
05000	PF2	Meb1	Interni	14/01/02	3000
05220	MR3	ComRom	pubblici	14/01/02	12200

CodiceCliente	TotaleOrdini
Ever34	8750
Meb1	6300
ComRom	23500

Qui sopra è rappresentata la tabella virtuale 'OrdiniPerCliente' su cui di potranno operare nuove queries:

```
SELECT CodiceCliente,TotaleOrdini
FROM OrdiniPerCliente
WHERE TotaleOrdini > 8000
```

Se volessimo sapere il numero medio di periti per zona, potremmo operare in questo modo:

CodicePerito	CognomeNome	Zona
PF2	Peruzzo Federico	Verona
MR3	Mario Rossi	Verona
BF2	Bonin Francesco	Brescia
CF3	Carlo Fonin	Venezia
MG1	Marco Grigoletto	Verona
AB1	Antonio Baggio	Brescia

```
CREATE VIEW PeritiPerZona (Zona, NumeroPeriti)
AS SELECT Zona, COUNT( * )
FROM Periti
GROUP BY Zona
```

Zona	NumeroPeriti
Verona	3
Brescia	2
Venezia	1

```
SELECT AVG(NumeroPeriti)
FROM PeritiPerZona
```

AVG
2

```
DROP PeritiPerZona
```

In media abbiamo 2 periti per zona.

Zona	NumeroPeriti
Verona	3
Brescia	2
Venezia	1

Con DROP elimino la view.

Esempio Completo di una Base di Dati:

Clienti(CodiceCliente, CongnomeNome, Città, Sconto)
Periti(CodicePerito,CognomeNome,Zona,Commissione)
Ordini(NumOrdine,CodiceCliente*,CodicePerito*,Prodotto,Data,Ammontare)

CodiceCliente chiave esterna per Clienti
CodicePerito chiave esterna per Periti

Codice SQL:

```
CREATE TABLE Clienti
      (
      CodiceCliente CHAR (5) UNIQUE NOT NULL,
      CognomeNome CHAR (30) NOT NULL,
      Città CHAR (30) NOT NULL,
      Sconto INTEGER NOT NULL CHECK (Sconto>0 AND Sconto<100)
      )
PRIMARY KEY pk.Clienti (CodiceCliente)

CREATE TABLE Periti
      (
      CodicePerito CHAR (5) UNIQUE NOT NULL,
      CognomeNome CHAR (30) NOT NULL,
      Zona CHAR (15) NOT NULL,
      Commissione INTEGER
      )
PRIMARY KEY pk.Periti (CodicePerito)

CREATE TABLE Ordini
      (
      NumOrdine CHAR (8) NOT NULL,
      CodiceCliente CHAR (5) NOT NULL,
      CodicePerito CHAR (5) NOT NULL,
      Prodotto CHAR (5) NOT NULL,
      Data CHAR (8) NOT NULL,
      Ammontare INTEGER NOT NULL CHECK (Ammontare >=100)
      )
PRIMARY KEY pk.Ordini (NumOrdine)
FOREIGN KEY fk.Ordini (CodiceCliente)
      REFERENCES Clienti
      ON DELETE NO ACTION
FOREIGN KEY fk.Ordini (CodicePerito)
      REFERENCES Periti
      ON DELETE NO ACTION

CREATE VIEW OrdiniPerCliente (CodiceCliente, TotaleOrdini)
AS SELECT CodiceCliente, SUM(Ammontare)
FROM Ordini
GROUP BY CodiceCliente

SELECT CodiceCliente,MAX(TotaleOrdini)
FROM OrdiniPerCliente
GROUP BY CodiceCliente
```

PRIMARY KEY definisce le chiavi primarie di ogni tabella, FOREIGN KEY le chiavi esterne; a questo ultimo operatore deve seguire REFERENCES che definisce la tabella a cui si riferisce la chiave esterna.
CHECK ci permette di definire con una condizione i valori ammissibili di un attributo.
E' possibile anche definire un attributo di default con la sintassi DEFAULT (costante | NULL).
UNIQUE definisce come unico il valore dell'attributo, quindi non ci possono essere duplicati: l'attributo è chiave.

TRIGGER

I *trigger* sono procedure che vengono attivate automaticamente in una base di dati quando si eseguono determinate operazioni (INSERT, UPDATE, DELETE).

```
CREATE TRIGGER NomeTrigger
      TipoTrigger
      (
      TipoOperazione {TipoOperazione}
      )
      [OF Attributo] ON NomeTabella
```

```
[FOR EACH ROW]
[WHEN "(" Condizione ")"]
Programma
```

- TipoTrigger: stabilisce quando il trigger debba essere attivato, BEFORE o AFTER.
- TipoOperazione e NomeTabella: definisce l'operazione che attiva il trigger e dove questa viene eseguita (eventualmente si possono inserire uno o più attributi)
- FOR EACH ROW: definisce il numero di volte in cui il trigger deve essere attivato (una volta o tante volte quante sono le righe della tabella interessate)
- Condizione: un'eventuale ulteriore condizione che deve essere vera perché il trigger venga attivato
- Programma: il codice da eseguire

ESERCIZI VARI:

1. Dato il seguente schema relazionale:

Studenti(Nome, Matricola, AnnoNascita, AnnoDiCorso)
EsamiSostenuti(Materia, Matricola, Giorno, Mese, Anno, Voto, Lode)

A) Trovare il nome degli studenti che hanno sostenuto l'esame di "Basi di Dati" con voto >25
B) Trovare quanti studenti hanno sostenuto "Basi di Dati" con voto pari a 30
C) Produrre una tabella con due colonne: Materia e Numero studenti che hanno sostenuto l'esame in quella materia

```
SELECT s.Nome
FROM Studenti s, EsamiSostenuti e
WHERE s.Matricola = e.Matricola AND e.Materia = "BD" AND e.Voto>25

SELECT COUNT(*)
FROM EsamiSostenuti
WHERE Materia = "BD" AND Voto = 30

SELECT Materia, COUNT(*) AS NumeroStudenti
FROM EsamiSostenuti
GROUP BY Materia
```

2. Dato il seguente schema relazionale:

Dentisti (Nome, Stanza)
Clienti (Codice, Nome, Indirizzo, Telefono)
Appuntamenti (NomeDentista, CodiceCliente, Data, OraInizio, Durata, TipoDiCura)

A) Trovare il numero di clienti che non hanno appuntamenti
B) Trovare la stanza il cui dentista ha il maggior numero di appuntamenti il 13/6/2003
C) Per ogni dentista dare il nome e il numero degli appuntamenti per il 15/09/2003

```
SELECT COUNT(*)
FROM Clienti c
WHERE NOT EXISTS
    (
    SELECT *
    FROM Appuntamenti a
    WHERE c.Codice=a.CodiceCliente
    )


CREATE VIEW TotaleAppuntamenti
SELECT NomeDentista, COUNT(*) AS Totale
FROM Appuntamente
WHERE Data = '13062003'
GROUP BY NomeDentista

SELECT d.Stanza
FROM Dentista d, TotaleAppuntamenti t
WHERE d.Nome = t.NomeDentista
AND t.Totale > ALL
    (
    SELECT Totale
    FROM TotaleAppuntamenti
    )


SELECT NomeDentista, COUNT(*) AS Numero Appuntamenti
```

```
FROM Appuntamenti
WHERE Data = '15092003'
GROUP BY NomeDentista
```

3. Dato il seguente schema di relazione:

Nazione (Nome, Valuta)
Clienti (Codice, Nazione, Nome, Indirizzo)
Ordini (Codice, CodiceCliente, CodiceProdotto, Quantità, Data)
Prodotti (Codice, Descrizione)

A) Trovare il codice di tutti gli ordini emessi da clienti francesi
B) Dare la descrizione del prodotto (o dei prodotti) con il più gran numero di ordini di clienti tedeschi
C) Cancellare tutti gli ordini dei clienti con valuta "Scellino"
D) Trovare codice e descrizione del prodotto con la quantità più venduta in assoluto

```
SELECT DISTINCT o.Codice
FROM Ordini o, Clienti c
WHERE c.Codice = o.CodiceCliente AND c.Nazione = 'Francia'


CREATE VIEW OrdiniDiTedeschi (CodiceOrdine)
AS SELECT o.Codice
FROM Ordini o, Clienti c
WHERE c.Codice = o.CodiceCliente AND c.Nazione = 'Germania'

CREATE VIEW ProdottiDiTedeschi (CodiceProdotto, Descrizione, NumeroOrdini)
AS SELECT p.Codice, p.Descrizione, COUNT(*) AS NumeroOrdini
FROM Prodotti p, OrdiniPerTedeschi o
WHERE o.CodiceProdotto = p.Codice
GROUP BY p.Codice, p.Descrizione

SELECT Descrizione
FROM ProdottiDiTedeschi
WHERE NumeroOrdini =
    (
    SELECT MAX(NumeroOrdini)
    FROM ProdottiDiTedeschi
    )


DELETE Ordini
WHERE CodiceCliente IN
    (
    SELECT c.Codice,
    FROM Clienti c, Nazioni n
    WHERE c.Nazione = n.Nome AND n.Valuta = 'Scellino'
    )


CREATE VIEW ProdottoQuantità (CodiceProdotto, Descrizione, Quantità)
AS SELECT p.Codice, p.Descrizione, SUM(o.Quantità)
FROM  Prodotti p, Ordini o
WHERE o.CodiceProdotto = p.Codice
GROUP BY p.Codice, p.Descrizione

SELECT CodiceProdotto, Descrizione
FROM ProdottoQuantità
WHERE Quantità =
    (
    SELECT MAX(Quantità)
    FROM ProdottoQuantità
    )
```

4. Dato il seguente schema di relazione:

Operai (Nome, Telefono, Indirizzo, CapoCantiere, NomeCantiere)
Cantieri (Nome, Città)
Lavori (NomeCantiere, NomeCapoCantiere, TipoLavoro, DataInizio, DataTermine)

A) Trovare il massimo numero di operai che lavorano in un cantiere
B) Trovare quanti lavori sono iniziati il 14/05/2003 nelle città di Verona, Brescia e Milano
C) Spostare la DataTermine al 15/09/2003 di tutti i lavori iniziati il 13/05/2003 e il 14/05/2003

```
CREATE VIEW OperaiPerCantiere (NomeCantiere, NumeroOperai)
AS SELECT c.Nome, COUNT(*)
FROM Operai o, Cantieri c
WHERE o.NomeCantiere = c.Nome
GROUP BY c.Nome

SELECT MAX(NumeroOperai)
FROM OperaiPerCantiere


SELECT COUNT(*)
FROM Lavori l, Cantieri c
WHERE c.Nome = l.NomeCantiere
      AND l.DataInizio = '14052003'
      AND c.Città IN ('Verona', 'Brescia', 'Milano')


UPDATE LAVORI
SET DataTermine = '15092003'
WHERE DataInizio IN ('13052003','14052003')
```

5. Dato il seguente schema di relazione:

Musei (Codice, Nome, Indirizzo)
Opere (Codice, Nome, Stile, CodiceMuseo, NomeAutore)
Autori (Nome, Periodo)

A) Trovare il nome del museo che ha più opere
B) Trovare il numero medio di opere esposte per autore al "Louvre" di Parigi
C) Trovare il nome dello stile che rappresenta meno opere agli "Uffizzi" di Firenze

```
CREATE VIEW OperePerMuseo (NomeMuseo, NumeroOpere)
AS SELECT m.Nome, COUNT(*)
FROM Musei m, Opere o
WHERE m.Codice = o.CodiceMuseo
GROUP BY m.Nome

SELECT NomeMuseo
FROM OperePerMuseo
WHERE NumeroOpere =
      (
      SELECT MAX(NumeroOpere)
      FROM OperePerMuseo
      )


CREATE VIEW OperePerAutoriLouvre (NomeAutore, NumeroOpere)
AS SELECT a.Nome, COUNT(*)
FROM Autori a, Opere o, Musei m
WHERE a.Nome = o.NomeAutore
       AND p.CodiceMuseo = m.Codice
       AND m.Nome = 'Louvre'
GROUP BY a.Nome

SELECT AVG(NumeroOpere)
FROM OperePerAutoriLouvre


CREATE VIEW OpereStiliUffizzi (NomeOpera, Stile)
AS SELECT o.Nome, o.Stile
FROM Opere o, Musei m
WHERE m.Nome = 'Uffizzi'
      AND m.Codice = o.CodiceMuseo

CREATE VIEW NumOperePerStili (Stile, NumeroOpere)
AS SELECT Stile, COUNT(*)
FROM OpereStiliUffizzi
GROUP BY Stile

SELECT Stile
FROM NumOperePerStili
WHERE NumeroOpere =
      (
      SELECT MIN(NumeroOpere)
      FROM NumOperePerStili
      )
```

6. Dato il seguente schema di relazione:

Reparti (Nome, CapoReparto, CapoInfermiere)
Medici (Nome, Indirizzo, Telefono, OraTurno, specializzazione, NomeReparto)
Emergenza (Ora, NomeMedico, NomeReparto)

A) Trovare il numero di medici che non lavorano al reparto di "Otorinolaringoiatria"
B) Creare una tabella che mostri chi è di turno per le emergenze alle ore 00:00 per i reparti di "Ginecologia", "Oncologia" e "Rianimazione"
C) Aumentare di 1 ora l'orario del turno di tutti i chirurghi

```
SELECT COUNT(*)
FROM Medici m
WHERE NOT EXISTS
    (
    SELECT *
    FROM Reparti r
    WHERE r.Nome = 'Otorinolaringoiatria' AND r.Nome = m.NomeReparto
    )
```

```
CREATE VIEW MediciInteressati (NomeMedico, Reparto, Turno)
AS SELECT Nome, NomeReparto, OraTurno
FROM Medici
WHERE NomeReparto IN ('Ginecologia', 'Oncologia', 'Rianimazione')

CREATE VIEW Medici00.00 (NomeMedico, Reparto)
AS SELECT NomeMedico, Reparto
FROM MediciInteressati
WHERE Turno = 00.00
```

```
UPDATE Medici
SET OraTurno = OraTurno + 60
WHERE Specializzazione = 'Chirurgia'
```

7. Dato il seguente schema di relazione:

Voli (Numero, oraPartenza, oraArrivo, Destinazione, TipoAereo)
Aerei (Tipo, Velocità, Capienza)
Prenotazioni (NumeroVolo, NomeCliente)

A) Trovare il tipo Aereo che ha una capienza maggiore rispetto al Boeing747
B) Trovare i voli (se ci sono) che hanno più prenotazioni rispetto alla capienza massima dell'aereo

```
SELECT Tipo
FROM Aerei
WHERE Capienza > ALL
    (
    SELECT Capienza
    FROM Aerei
    WHERE Tipo = 'Boeing747'
    )
```

```
CREATE VIEW PrenotazioniPerVoli (NomerVolo, NumeroPrenotazioni, TipoAereo)
AS SELECT p.NumeroVolo, COUNT(*), v.TipoAereo
FROM Prenotazioni p, Voli v
WHERE p.NumeroVolo = v.Numero
GROUP BY p.NumeroVolo

SELECT a.Tipo
FROM Aerei a, PrenotazioniPerVoli pp
WHERE a.Tipo = pp.TipoAereo
    AND pp.NumeroPrenotazioni > a.Capienza
```

8. Dato il seguente schema di relazione:

Sale (Numero, Capienza, Film)
Sportelli (Numero, NomeFilm, Prenotazioni, Data)
Film (Nome, Tempo, NumeroSala)

A) Trovare il numero dello sportello che ha effettuato più prenotazioni per il film "Una settimana da dio"

B) Mostrare i nomi di tutti i film le cui prenotazioni sono pari alla capienza delle sale corrispondenti nel giorno 12/05/2003
C) Trovare il numero della sala e il nome del film proiettato in cui nel giorno 12/05/2003 c'è stata minor affluenza

```
SELECT sp.Numero
FROM Sportelli sp, Film f
WHERE f.Nome = "Una settimana da Dio"
        AND f.Nome = sp.NomeFilm
        AND sp.Prenotazioni =
        (
        SELECT MAX(Prenotazioni)
        FROM Sportello
        )
```

```
CREATE VIEW  FilmPrenota12.05.2003 (NomeFilm, NumeroPrenotazioni)
AS SELECT NomeFilm, SUM(Prenotazioni)
FROM Sportelli
WHERE Data = '12052003'
GROUP BY NomeFilm

 SELECT fp.NomeFilm
FROM FilmPrenota12.05.2003, Sale s
WHERE fp.NomeFilm = s.Film
        AND s.Capienza = fp.NumeroPrenotazioni
```

```
CREATE VIEW SaleFilmPrenot (NumeroSala, NomeFilm, NumeroPrenot)
AS SELECT s.Numero, sp.NomeFilm, SUM(Prenotazioni)
FROM Sale, Sportelli
WHERE sp.NomeFilm = s.Film
        AND sp.Data = '12052003'
GROUP BY s.Numero, sp.NomeFilm

SELECT NumeroSala, NomeFilm
FROM SaleFilmPrenot
WHERE NumeroPrenot =
        (
        SELECT MIN(NumeroPrenot)
        FROM SaleFilmPrenot
        )
```

9. Dato il seguente schema relazionale:

CentriCommerciali (Nome, Indirizzo, NumNegozi, Città)
Negozi (Nome, Piano, NomeResposabile, NomeCentro)
Lavoratori (Nome, Indirizzo, NomeNegozio)

A) Trovare il nome del centro commerciale e del negozio dove lavora il Signor Mario Rossi che abita in Via G.Verdi
B) Creare una tabella che mostri il numero dei negozi per ogni piano. Un piano per essere visualizzato deve avere più di due negozi
C) Trovare il nome e il relativo indirizzo del centro commerciale che ha più negozi al terzo piano rispetto ai Centri Commerciali di Roma

```
SELECT n.NomeCentro, n.Nome
FROM Negozi n, Lavoratori l
WHERE l.Nome = 'Mario Rossi'
        AND l.Indirizzo = 'Via G.Verdi'
        AND l.NomeNegozio = n.Nome
```

```
CREATE VIEW NegoziPerPiano (NumeroPiano, NumeroNegozi)
AS SELECT Piano, COUNT(*)
FROM Negozi
GROUP BY Piano
HAVING COUNT(*) > 2
```

```
CREATE VIEW Centro3Piano (NomeCentro, Negozi3Piano)
AS SELECT n.NomeCentro, COUNT(*)
FROM Negozi n, CentriCommerciali c
WHERE n.Piano = 3
        AND c.Nome = n.NomeCentro
GROUP BY n.NomeCentro
```

```
SELECT cp.NomeCentro, c.Indirizzo
FROM Centro3Piano cp, CentriCommerciali c
WHERE cp.NomeCentro = c.Nome
        AND cp.Negozi3Piano > ALL
        (
        SELECT Negozi3Piano
        FROM Centro3Piano
        WHERE Città = 'Roma'
        )
```

10. Dato il seguente schema relazionale:

Discoteche (Nome, Indirizzo, Città, Capienza)
Performance (Tipo, Data, NomeOspite, NomeDisco)
Ospiti (Nome, Recapito, Agente, TipoSpettacolo)

A) Creare una tabella che mostri quali Concerti sono previsti per il giorno 02/09/2003 nelle discoteche di Verona
B) Trovare il nome della/e Discoteca/he in cui non è mai stato invitato un cabarettista.
C) Cancellare il concerto dal vivo previsto alla Discoteca "Freedom" per il giorno 01/10/2003

```
CREATE VIEW DiscoVerona (NomeDiscoteca)
AS SELECT d.Nome
FROM Discoteche d, Performance p
WHERE d.Nome = p.NomeDisco
        AND d.Città = 'Verona'

CREATE VIEW DiscoConcerti29 (NomeDiscoteca, Concerto)
AS SELECT dv.NomeDiscoteca, p.NomeOspite
FROM DiscoConcerti29 dv, Performance p
WHERE dv.NomeDiscoteca = p.NomeDisco
        AND p.Data = '02092003'
        AND p.Tipo = 'Concerto'


SELECT d.Nome
FROM Discoteche d, Performance p
WHERE d.Nome = p.NomeDisco
        AND NOT EXISTS
        (
        SELECT *
        FROM Performance
        WHERE Tipo = 'Cabaret'
        )


DELETE
FROM Performace
WHERE NomeDisco = 'Freedom'
        AND Data = '01102003'
        AND Tipo = 'Concerto'
```

NORMALIZZAZIONE

Per schematizzare una Base di Dati, esistono di solito diverse rappresentazioni possibili. Il problema consiste nel verificare se queste diverse rappresentazioni sono tra loro equivalenti. E' quindi necessario valutare se esistano delle *anomalie* (perdita di informazioni, ripetizioni di informazioni, impossibilità di rappresentare certi fatti…).
La teoria della normalizzazione si occupa di:

- definire quando due schemi sono equivalenti
- definire criteri di bontà per schemi (qual è il migliore e perché)
- trovare metodi algoritmici per ottenere da uno schema uno schema migliore ed equivalente

DIPENDENZE FUNZIONALI

→ è un vincolo d'integrità tra un insieme di attributi X e un insieme di attributi Y di uno stesso schema di relazione e si esprime con X → Y (X determina Y) tale che:

$$X \rightarrow Y \Leftrightarrow \forall\ t1,t2 \in r,\ t1[X] = t2[X] \Rightarrow t1[Y] = t2[Y]$$

Magazzino (CodiceProdotto, DescrizioneProdotto, PrezzoProdotto)

CodiceProdotto → DescrizioneProdotto, PrezzoProdotto

Videoteca (CodiceVideo, NomeVideo, TipologiaVideo, Data, CodiceCliente, NomeCliente, ResidenzaCliente, TelefonoCliente)

CodiceVideo → NomeVideo, TipologiaVideo
CodiceCliente → NomeCliente, ResidenzaCliente, TelefonoCliente
CodiceVideo → CodiceCliente, Data

ASSIOMI DI ARMSTRONG

Riflessività: se $Y \subseteq X$, allora X → Y
Arricchimento: se X → Y e $W \subseteq T$ allora XW → YW
Transitività: se X → Y e Y → Z, allora X → Z

COPERTURA CANONICA

Un insieme di dipendenze F è una copertura canonica se e solo se:

- ogni parte destra di una dipendenza ha un solo attributo
- le dipendenze non contengono attributi estranei
- non esistono dipendenze ridondanti

→ attributo estraneo: dato X → Y ∈ F, X contiene un attributo estraneo A se $(F - \{X \rightarrow Y\}) \cup \{X - \{A\} \rightarrow Y\} \equiv F$

→ X → Y è una dipendenza ridondante se e solo se $(F - \{X \rightarrow Y\}) \equiv F$

TERZA FORMA NORMALE

Un insieme di dipendenze F è in 3FN se e solo se per ogni dipendenza funzionale non banale $X \rightarrow A \in F^+$, allora X è una superchiave o A è primo.
Esempio:
R<(A,B,C,D), (AB→C, C→D)>
AC è chiave.
La DF A→D rispetta la prima condizione del teorema (A è chiave)
La DF C→B rispetta la prima condizione, perché C è attributo primo

Algoritmo:

1. trovare una copertura canonica G di F
2. sostituire in G ogni insieme X→A1,….X→An di dipendenze con lo stesso determinantem con la dipendenza X→A1,…,An
3. per ogni dipendenza X→Y in G, si metta nella decomposizione p uno schema con attributi XY

4. si eliminino dalla decomposizione p schemi che sono già contenuti in altri schemi
5. se la decomposizione p non contiene uno schema i cui attributi sono una superchiave di R, si aggiunge lo schema con attributi W, con W chiave di R

Esempio
R ({A,B,C,D,E,F,G},{BDG→AC, CD→F, A→E, FE→ABD, F→G})

2) Copertura canonica

a-
Ogni parte destra di una dipendenza deve essere costituita da un solo attributo

BDG→A
BDG→C
CD→F
A→E
FE→A
FE→B
FE→D
F→G

b-
Le dipendenze non devono contenere attributi estranei

BDG→A
{DG}+ = DG B non è estraneo
{BG}+ = BG D non è estraneo
{BD}+ = BD G non è estraneo
…….

c-
Non devono esserci dipendenze ridondanti

Togliendo BDG→A
{BDG}+ = BDGCF non ottengo A, non ridondante

Togliendo BDG→C
{BDG}+ = BDGAE non ottengo C, non ridondante

Togliendo CD→F
{CD}+ = CD non ottengo F, non ridondante

Togliendo A→E
{A}+ = A non ottengo E, non ridondante

Togliendo FE→A
{FE}+ = FEBDGC**A** ottengo A (BDG→A), è ridondante

Togliendo FE→B
{FE}+ = FEDG non ottengo B, non ridondante

Togliendo FE→D
{FE}+ = FEBG non ottengo D, non ridondante

Togliendo F→G
{F}+ = F non ottengo G, non ridondante

3) Trovare una chiave

{FE}+ = FEBDGCA → FE è chiave

{BDG}+ = BDGACFE → BDG è chiave

4) Dire se la relazione è in 3FN

BDG→A, BDG è superchiave
BDG→C, BDG è superchiave
CD→F, F è primo
A→E, E è primo
FE→B, FE è superchiave
FE→D, FE è superchiave
F→G, G è primo

(FE→A non è stata considerata perché dipendenza ridondante)

La relazione R è in 3FN.

Esempio

R ({A,B,C,D,E,F,G },{AB→CD, AB→E , C→F, F→G})

AB→CD viene decomposto in
AB→C
AB→D
Non ci sono attributi estranei

AB→C = ABDE non ottengo C
AB→D = ABCEFG non ottengo D
AB→E = ABCDFG non ottengo E
C→F = C non ottengo F
F→G = F non ottengo G
Non ci sono dipendenze ridondanti
E' in forma canonica
La chiave è AB (ABCDEFG)

C→F e F→G non rispettano le condizioni descritte nella terza forma normale.
E' necessario normalizzare R.

R1 <ABCDE>
R2 <CF>
R3 <FG>

Esempio

R ({A,B,C,D},{A→BC, B→A , C→D})

A→B
A→C

Non ci sono attributi estranei.

A→C = AB non ottengo C
A→B = ACD non ottengo B
B→A = B ottengo A
C→D = C non ottengo D
E' in forma canonica
A è chiave.
Non è in forma normale per C→D

R1 <ABC>
R2 <CD>

A→BC e B→A sono state fuse assieme in R1 perchè si determinano reciprocamente. (più chiavi alternate per lo stesso schema).

Esempio

R ({A,B,C,D },{A→C, B→D })

Non ci sono attributi estranei o ridondanze.
E' in forma canonica.
AB è chiave, ma nessuna dipendenza rispetta le direttive della terza forma normale.
Normalizziamo:

R1 <AC>
R2 <BD>

Ma non basta. Abbiamo bisogno anche di:

R3 <AB>, ovvero lo schema che ha come chiave quella originale di R (appunto AB)

Esempio

R ({A,B,C,D,E,F,G,H },{DEF→ABC, AC→G, GC→E })

DEF→A
DEF→B
DEF→C
AC→G
GC→E

Non ci sono attributi estranei.
Non ci sono dipendenze ridondanti.
DEFH è chiave.
Non è in 3FN per:
AC→G

R1 <DEFABC>
R2 <ACG>
R3 <GCE>
R4 <DEFH> (da aggiungere per preservare i dati)

FORMA NORMALE BOYCE-CODD

L'idea su cui si basa la nozione di FNBC è che una dipendenza funzionale X→Y indica che una collezione C di entità è univocamente identificata da X.
A questo punto o X è una chiave, o non deve apparire nessuna relazione X→A (in questo ultimo caso, se X non è chiave, X è chiave esterna e quindi referente di un'altra relazione).
Quindi: Uno schema R(T,F) è in FNBC se per ogni dipendenza X→Y ∈ F+, X è una superchiave.
Si può quindi dire che se una relazione è in FNBC è anche in 3FN.

Esempio

R({A,B,C,D,E},{A→B,DB→C,BE→A})

Ogni parte destra è già costituita da un solo attributo.
Non ci sono attributi estranei.
{A }+= A non ottengo B, non ridondante
{DB}+=DB non ottengo C, non ridondante
{BE}+= BE non ottengo A, non ridondante

Le dipendenze sono una copertura canonica.
Una chiave può essere DBE.
Nessun determinante è una superchiave, quindi non può essere in FNBC.

A→B

R1<AB, {A→B}>
R2<ACDE, {}>
ACDE è superchiave

DB→C

R1<DBC, {DB→C }>
R2<ABDE, {A→B, BE→A}>
R3a<ADE, {}>
R3b<BDE, {}>
BDE è superchiave (e chiave)

BE→A

R1<BEA, {BE→A}>
R2<BCDE, {DB→C}>
R3<BDE, {}>
BDE è superchiave (e chiave)

Decomposizione in FNBC effettuata.

Esempio

R({A,B,C,D,E,F},{AB→C,CD→F,FB→E})

Tutte le dipendenze hanno un solo attributo a destra.

Non ci sono attributi estranei.
{AB}+ = AB, non ottengo C, non ridondante
{CD}+ = CD, non ottengo F, non ridondante
{FB}+ = FB, non ottengo E, non ridondante

Le dipendenze sono una copertura canonica.
Un esempio di chiave è ABD (ABDCFE).
Nessun determinante è una superchiave, quindi R non è in FNBC.

Decomposizione:

AB→C

R1<ABC, {AB→C}>
R2<ABDEF, {FB→E}>
R3<ABDF, {}>
ABDE è superchiave

CD→F

R1<CDF, {CD→F}>
R2<ABCDE, {AB→C}>
R3<ABDE, {}>
ABDE è superchiave

FB→E

R1<FBE, {FB→E}>
R2<ABCDF, {AB→C,CD→F}>
R3a<ABDF, {}>
R3b<ABCD, {AB→C}>
R4b<ABD, {}>
ABD è superchiave (e chiave)

Decomposizione in FNBC effettuata.

Esempio

R({A,B,C,D,E,F,G},{AB→CFG,CB→ACE,CB→DF,A→DG})

AB→C
AB→F
AB→G
CB→A
CB→C
CB→E
CB→D
CB→F
A→D
A→G

Elementi estranei

AB→C
{A}+ = ADG
{B}+ = B

AB→F
{A}+ = ADG
{B}+ = B

AB→G
{A}+ = ADG, B estraneo
{B}+ = B

CB→A
{C}+ = C
{B}+ = B

CB→C
{C}+ = C, B estraneo

CB→E
Nessun estraneo

CB→D
Nessun estraneo

CB→F
Nessun estraneo

A→D
Nessun estraneo

A→G
Nessun estraneo

Dipendenze Ridondanti

AB→C = ABCFGDE
AB→F = ABFCEDGF, ridondante
CB→A = CBAEDF
CB→E = CBEADF
CB→D = CBDAEFGD, ridondante
CB→F = CBFDEAG
A→D = ADG
A→G = AGD

AB e BC sono chiavi.

Non è in FNBC per A→D e A→G

A→D
R1 <(AD), (A→D)>
R2 <(ABCEFG), (AB→C, CB→EF, A→G)>
R3a <(ABEFG), (A→G)>
R4a <(ABEF), ( )>
R3b <(ABCG), (AB→C, A→G)>
R4c <(ABG), (A→G)>
R5c <(AB), ()>
R4d <(ABC), (AB→C)>
R5d <(AB), ()>
ABEF è superchiave e AB è chiave

A→G
R1 <(AG), (A→G)>
R2 <(ABCDEF), (AB→C, CB→EF)>
R3a <(ABEFG), (A→G)>
R4a <(ABEF), ( )>
R3b <(ABCD), (AB→C, A-->D)>
R4c <(ABD), (A-->D)>
R5c <(AB), ()>
R4d <(ABC), (AB-->C)>
R5d <(AB), ()>
ABEF è superchiave e AB è chiave