

Basi di Dati in Pillole

Sofia Amarù

31 agosto 2019

Contents

1	Introduzione	5
1.1	Concetti fondamentali	5
1.1.1	Sistema informativo, sistema informatico, dati	5
1.1.2	Definizione di database, DBMS	5
1.1.3	Caratteristiche	5
1.1.4	Modelli	6
1.1.5	Transizioni, schema, istanza	6
1.1.6	Livelli di astrazione	6
1.1.7	Indipendenza dei dati	6
1.1.8	Linguaggi	6
2	Modello ER (entità-relazione)	7
2.1	Rappresentazione grafica dei costrutti	7
2.2	Costrutti	7
2.3	Esercizi	8
2.3.1	Esercizio 1	8
2.3.2	Esercizio 2	8
3	Modello Relazionale	10
3.1	Vincoli di integrità	10
3.2	Chiave, superchiave, superchiave minimale	10
3.3	Esercizi	10
3.3.1	Esercizio 1	10
4	Progettazione logica	12
4.1	Ristrutturazione schema ER	12
4.1.1	Analisi delle ridondanze	12
4.1.2	Eliminazione delle generalizzazioni	12
4.1.3	Partizionamento	12
4.1.4	Eliminazione di attributi multivalore	13
4.1.5	Accorpamento di entità	13
4.1.6	Scelta degli identificatori principali	13
4.2	Traduzione verso modello logico	13
4.2.1	Molti a molti	13
4.2.2	Uno a molti	13
4.2.3	Uno a uno con partecipazioni obbligatorie per entrambe le entità	14
4.2.4	Uno a uno con partecipazione opzionale per una sola entità	14
4.2.5	Attributi opzionali	14
4.3	Esercizi	14
4.3.1	Esercizio 1	14
5	Algebra relazionale	16
5.1	Operatori insiemistici	16
5.2	Operatori principali	16
5.2.1	Ridenominazione	16
5.2.2	Selezione	16

5.2.3	Proiezione	16
5.2.4	Join	17
5.2.5	Prodotto cartesiano	17
5.2.6	Theta join e equi-join	17
5.2.7	Logica a tre valori	18
5.2.8	Interrogazioni	18
5.3	Esercizi	18
5.3.1	Esercizio 1	18
5.3.2	Esercizio 2	19
5.3.3	Esercizio 3	20
6	SQL	21
6.1	Operatori	21
6.1.1	Operatori di assegnazione	21
6.1.2	Operatori di confronto	22
6.1.3	Operatori aritmetici	22
6.1.4	Operatori condizionali	22
6.1.5	Operatori logici	23
6.2	Interrogazioni	23
6.2.1	Select, from, where	23
6.2.2	Is null	24
6.2.3	Distinct, all	25
6.2.4	Join, alias	25
6.2.5	Order by	25
6.2.6	Operatori aggregati	26
6.2.7	Group by	26
6.2.8	Predicati su gruppi: having	26
6.2.9	Interrogazioni di tipo insiemistico	26
6.2.10	Any	27
6.2.11	Sintassi completa	27
6.3	Viste	27
6.3.1	Esempio 1	27
6.3.2	Esempio 2	27
6.4	Esercizi	28
6.4.1	Esercizio 1	28
6.4.2	Esercizio 2	30
6.4.3	Esercizio 3	30
6.4.4	Esercizio 4	31
6.4.5	Esercizio 2	34
6.4.6	Esercizio 3	34
6.4.7	Esercizio 4	35
6.4.8	Esercizio 5	36
7	Link utili	39

Note

Il seguente testo è un personalissimo specchietto riassuntivo dei concetti fondamentali studiati durante il corso di Basi di Dati (corso di laurea triennale in Informatica presso l'Università degli Studi di Milano-Bicocca) e lo studio sullo stesso non è sufficiente per il superamento dell'esame. Si consiglia la lettura solo dopo aver studiato dal libro *Basi di Dati - Modelli e linguaggi di interrogazione* degli autori Azteni, Ceri, Paraboschi, Torlone.

Questo testo può contenere errori: nell'eventualità vi prego di contattarmi per procedere alla correzione.

Ricordo ai miei compagni che l'esame è composto da 5 esercizi relativi ai seguenti argomenti:

- Modello ER
- Modello Relazionale
- Progettazione logica
- SQL
- Algebra relazionale

1 Introduzione

1.1 Concetti fondamentali

1.1.1 Sistema informativo, sistema informatico, dati

sistema informativo: componente (non necessariamente automatizzata) di un'organizzazione, che gestisce le informazioni di interesse

sistema informatico: porzione automatizzata del sistema informativo

- acquisizione e memorizzazione
- aggiornamento
- interrogazione
- elaborazione

dato \neq informazione

- dato: non ha alcun valore fino a quando non viene interpretato (immutato nel tempo)
- informazione: si ha quando un dato viene interpretato

1.1.2 Definizione di database, DBMS

DATABASE: collezione di dati usati per rappresentare informazioni di interesse

DBMS (data base management system): software per la gestione di DB

1.1.3 Caratteristiche

caratteristiche di un DB:

- privacy
- affidabilità
- efficienza
- efficacia

fasi:

- definizione
- creazione e popolazione
- manipolazione

caratteristiche di un sistema di DB:

- natura autodescrittiva
- separazione tra programmi e dati
- astrazione dei dati (visione concettuale del DB per l'utente)
- supporto viste multiple dei dati
- condivisione dei dati e gestione con utenti multipli

1.1.4 Modelli

modello logico:

- relazionale (costruttore relazione; record a struttura fissa; tabella)
- gerarchico (strutture ad albero)
- reticolare (grafi)
- a oggetti
- XML

modello concettuale:

- ER (entità-relazione)

1.1.5 Transizioni, schema, istanza

transizione: insieme indivisibile di operazioni

schema: struttura, invariante nel tempo

istanza: valori attuali, possono cambiare anche molto rapidamente

1.1.6 Livelli di astrazione

schema logico - descrizione dell'intera base di dati per mezzo del modello logico

schema interno - rappresentazione dello schema logico per mezzo di strutture fisiche di memorizzazione

schema esterno - descrizione di una porzione della base di dati per mezzo del modello logico. A volte non è esplicitamente presente ma è possibile definire relazioni derivate (viste)

1.1.7 Indipendenza dei dati

indipendenza fisica - interagire col DBMS indipendentemente dalla struttura fisica dei dati

indipendenza logica - interagire col livello esterno indipendentemente dal livello logico

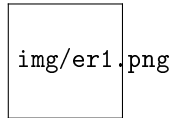
1.1.8 Linguaggi

DDL Data Definition Language - definizione degli schemi e delle autorizzazioni per l'accesso

DML Data Manipulation Language - interrogazione e aggiornamento delle istanze del DB

2 Modello ER (entità-relazione)

2.1 Rappresentazione grafica dei costrutti



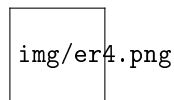
2.2 Costrutti

entità - classe di oggetti con proprietà comuni (il nome va al singolare)

relazione - legame logico (il nome deve essere univoco; è preferibile usare un sostantivo per evitare di assegnare un verso)

attributo composto - raggruppamento di attributi (Es. Via, Numero, CAP)

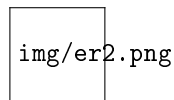
cardinalità di relazione - indica quante volte, in una relazione, l'occorrenza dell'entità è legata ad occorrenze dell'altra entità



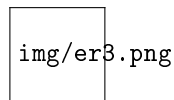
un impiegato ha da 1 a 5 incarichi

identificatore (chiave) - identificatore univoco dell'entità

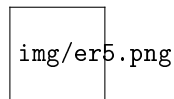
identificatore interno - se è uno o più attributi di un'entità



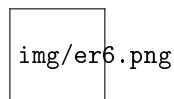
identificatore esterno - se un'entità viene identificata da un attributo di un'altra entità con cui ha una relazione



generalizzazione totale - non esistono altri figli



generalizzazione parziale - ci possono essere altri figli



sottoinsieme - un solo figlio

generalizzazione esclusiva - es. uomo-donna (o è uomo o è donna)

generalizzazione sovrapposta - es. studente-lavoratore (può essere studente, lavoratore o studente-lavoratore)

2.3 Esercizi

2.3.1 Esercizio 1

Una grande città vuole rappresentare una parte del territorio in una base dati. Anzitutto vuole rappresentare le attuali linee metropolitane, ciascuna con un id (linea 1, linea 2, ecc.), un colore e una lunghezza in km. Ad ogni linea sono associate delle fermate, ciascuna identificata da un numero progressivo nell'ambito della linea (fermata 1 della linea 3, fermata 2 della linea 2, fermata 3 della linea 3 ecc.), da un nome (es. Castello, Duomo, ecc.) e da una profondità rispetto al piano strada. Quindi occorre assumere che una fermata di una linea, in cui transitano ovviamente treni nelle due direzioni, deve essere rappresentata come un'unica fermata.

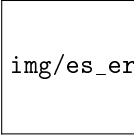
Inoltre, si vuole rappresentare per ogni fermata quale sia la fermata successiva nella stessa linea, e la distanza tra le due (es. la distanza tra le fermate 3 e 4 della linea 2 è di 700 metri).

Alcune coppie di fermate di due linee sono di incrocio tra due linee, in cui una linea passa «sotto» e l'altra passa «sopra»; si vuole rappresentare la relazione tra le due fermate (ad esempio, a piazzale Loreto della metro milanese si incrociano le linee rossa e verde. Questo significa che a Loreto corrispondono due distinte fermate, per esempio, la fermata 3 della linea rossa e la fermata 7 della linea verde, tra cui c'è una relazione di incrocio). È facoltativo rappresentare i casi in cui vi siano incroci tra tre fermate.

Si vogliono poi rappresentare gli sponsor delle fermate (ad esempio l'azienda Prysmian è sponsor della fermata 2 della linea 7), con nome dello sponsor e sua partita iva. Ogni sponsor può esserlo di più fermate, ogni fermata può avere più sponsor. Gli sponsor pagano cifre mensili diverse per le diverse fermate, e anche questo attributo deve essere rappresentato nella base di dati.

Infine si vogliono rappresentare i principali edifici della città, con codice e nome dell'edificio. Gli edifici sono di due tipi: musei, di cui si vuole rappresentare il costo del biglietto, e monumenti, di cui si vuole rappresentare l'orario di apertura. Per i musei pubblici, e solo per quelli pubblici, si vuole rappresentare il giorno settimanale di accesso libero (lunedì, martedì, ecc.). Per ogni edificio si vuole rappresentare la fermata della metropolitana più vicina, con la distanza a piedi.

Rappresentare tutto mediante il modello ER, con identificatori, cardinalità minime e massime. Rappresentare lo schema ER in modo esteticamente gradevole.

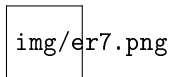


img/es_er.png

2.3.2 Esercizio 2

Il servizio di autoambulanze di una città deve organizzare un insieme di informazioni. Anzitutto ogni ambulanza è identificata da un codice, tipo automobile. Per ogni ambulanza si vuole poi rappresentare l'autorimessa di sosta, con via e numero civico, ed il personale assegnato, sempre lo stesso, con CF, nome, cognome, ruolo. Ogni chiamata è registrata con giorno, ora, minuto, persona

chiamante, con CF, nome e cognome, e persona da trasportare, con CF, nome e cognome. Se la persona da trasportare è minorenne occorre rappresentare l'età. Deve poi essere rappresentato l'insieme dei ponti soccorso, con il nome, e che possono avere varie specializzazioni cioè generale, oftalmico, traumatologico, ecc. Quando l'ambulanza interviene deve essere associato alla chiamata (e quindi alla persona trasportata) una prima diagnosi, con codice, classificazione e nome, che permetta di capire quali ospedali possono offrire cure immediate; come conseguenza deve essere permanentemente rappresentata una relazione tra diagnosi e specializzazione



3 Modello Relazionale

3.1 Vincoli di integrità

intrarelazionale - soddisfatto rispetto a singole relazioni

di tupla - può essere valutato su ciascuna tupla indipendentemente dalle altre.

Es. la lode può comparire solo con voto 30

di dominio - restrizione sul dominio dell'attributo

Es. voto compreso tra 18 e 30

interrelazionale - coinvolge più relazioni

vincolo di integrità referenziale - proprietà dei dati che per essere soddisfatta richiede che ogni valore di un attributo (colonna) di una relazione (tabella) esista come valore di un altro attributo in un'altra relazione.

Es. matricola compare in ESAMI solo se compare in STUDENTI

3.2 Chiave, superchiave, superchiave minimale

chiave - insieme di attributi utilizzato per identificare univocamente le tuple di una relazione

superchiave - insieme di attributi di una relazione tali che le relative tuple sono tutte diverse tra loro (identificazione univoca)


superchiave minimale - superchiave che non contiene una superchiave

chiave primaria - chiave a cui non è permesso contenere valori nulli (la presenza di NULL impedisce l'identificazione univoca)

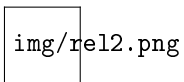
3.3 Esercizi

3.3.1 Esercizio 1

La seguente base di dati rappresenta l'archivio di una banca. La banca ha diverse filiali. I conti correnti possono avere un intestatario ed eventualmente un cointestatario. Alcuni clienti hanno più di un conto. Si tiene traccia anche di eventuali conti chiusi. I movimenti sui conti possono essere effettuati in filiale da un operatore, oppure via WEB, in questo caso non risulta alcun operatore associato a quel movimento. I clienti possono stipulare al massimo un mutuo in un dato giorno, ed effettuare al massimo un movimento in una data ora di un giorno.

 rel1.png

Definire tutte le chiavi primarie e i tutti i vincoli di integrità referenziale. Le chiavi primarie ed i vincoli di integrità referenziale possono essere espressi in modo grafico direttamente sul testo. I vincoli di integrità referenziale possono essere espressi in modo grafico.



Indicare:

1. un vincolo di dominio
importo > 0 in MOVIMENTICONTO
2. un vincolo di ennupla
dataApertura antecedente dataChiusura in CONTOCORRENTE
3. una superchiave non minimale
CF, nome, cognome in CLIENTE
4. una chiave che non sia stata scelta come chiave primaria
CF in PERSONALE
5. due attributi che possano assumere valore NULL
operatore in MOVIMENTICONTO;
dataChiusura in CONTOCORRENTE

4 Progettazione logica

Fasi

- **Ristrutturazione schema ER**
 - Analisi delle ridondanze
 - Eliminazione delle generalizzazioni
 - Partizionamento/accorpamento entità e associazioni
 - Scelta identificatori principali
- **Traduzione verso modello logico**

4.1 Ristrutturazione schema ER

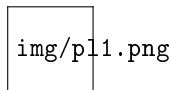
4.1.1 Analisi delle ridondanze

La presenza di un dato ridondante comporta:

- × occupazione memoria
- × operazioni per mantenere dato aggiornato
- ✓ riduzione accessi necessari per calcolarlo

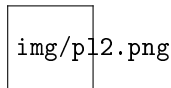
4.1.2 Eliminazione delle generalizzazioni

Accorpamento dei figli nel genitore Si usa quando l'accesso alle entità è contestuale.



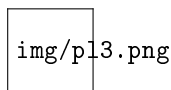
Accorpamento del genitore dei figli

NB. Possibile solo se la generalizzazione è totale!



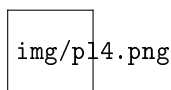
Sostituzione generalizzazioni con associazioni Si usa quando l'accesso alle due entità è separato.

NB. Conveniente con generalizzazione parziale



4.1.3 Partizionamento

decomposizione verticale



decomposizione orizzontale

img/p15.png

4.1.4 Eliminazione di attributi multivalore

Es. Un'agenzia può avere più numeri di telefono

img/p16.png

4.1.5 Accorpamento di entità

NB. Solitamente su associazioni uno a uno

img/p17.png

4.1.6 Scelta degli identificatori principali

- No attributi con valori nulli
- Preferire identificatore composto da 1 o **pochi** attributi
- Preferire identificatore interno ad identificatore esterno
- Preferire identificatore che viene utilizzato da molte operazioni per accedere alle occorrenze dell'entità

4.2 Traduzione verso modello logico

4.2.1 Molti a molti

img/p18.png

IMPIEGATO(Matricola, Cognome, Stipendio)

PROGETTO(Codice, Nome, Budget)

PARTECIPAZIONE(Impiegato, Progetto, DataInizio)

NB. Impiegato è la ridenominazione di Matricola e Progetto è la ridenominazione di Codice

4.2.2 Uno a molti

img/p19.png

GIOCATORE(DataNascita, Cognome, Ruolo)


SQUADRA(Nome, Città)

CONTRATTO(Giocatore, Squadra, Ingaggio)

diventa

GIOCATORE(DataNascita, Cognome, Ruolo, Squadra, Ingaggio)
SQUADRA(Nome, Città)

4.2.3 Uno a uno con partecipazioni obbligatorie per entrambe le entità




img/pl110.png

DIRETTORE(Matricola, Cognome, Stipendio, Dipartimento, InizioDirezione)
DIPARTIMENTO(Nome, Telefono, Sede)

oppure

DIRETTORE(Matricola, Cognome, Stipendio)
DIPARTIMENTO(Nome, Telefono, Sede, Direttore, InizioDirezione)

4.2.4 Uno a uno con partecipazione opzionale per una sola entità




img/pl111.png

IMPIEGATO(Matricola, Cognome, Stipendio)
DIPARTIMENTO(Nome, Telefono, Sede)
DIREZIONE(Direttore, Dipartimento, DataInizio)

4.2.5 Attributi opzionali

Gli attributi opzionali vengono contrassegnati con il simbolo *




img/pl112.png

CLIENTE(Codice, Nome, Indirizzo, CF*, P.iva*)

4.3 Esercizi


4.3.1 Esercizio 1

Dato il seguente schema Entità-Relazione, produrre lo schema ristrutturato e semplificato, tenendo presente che la grande maggioranza delle operazioni opera separatamente sulle entità E5 ed E4.



img/pl113.png

Soluzione:



img/pl114.png

E1(A11, A12, A13)
E2(A21, A22)
R1(A11, A21, A31)
R4(A21-1, A21-2)
E3(A31, A32, A51-1, A51-2)
~~R2()~~
E5(A51, A52)
~~R3()~~
E4(A51, A41)
~~R5()~~

5 Algebra relazionale

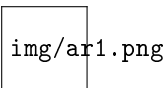
Relazione: insieme di tuple omogenee.

5.1 Operatori insiemistici

unione \cup

intersezione \cap

differenza $-$

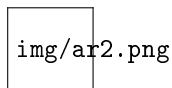


5.2 Operatori principali

5.2.1 Ridenominazione

Agisce solo sullo schema, cambiando solo i nomi degli attributi

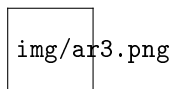
$\rho_{\text{nuovo nome} \leftarrow \text{vecchio nome}}$



5.2.2 Selezione

Produce un sottoinsieme delle tuple su tutti gli attributi (decomposizione orizzontale)

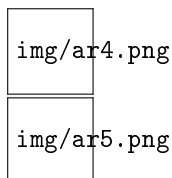
$\sigma_{\text{condizione di selezione}}$



5.2.3 Proiezione

È l'insieme delle tuple ottenute considerando solo i valori dei campi scelti. La proiezione contiene lo stesso numero di tuple della relazione di partenza solo se i campi scelti sono una superchiave. (decomposizione verticale)

$\pi_{\text{campi scelti}}$

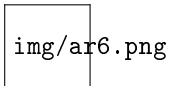


5.2.4 Join

Permette di correlare dati contenuti in relazioni diverse

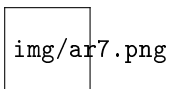
⋈ uguaglianza tra campi

join naturale - correla dati in relazioni diverse sulla base di valori uguali in campi uguali.



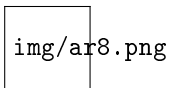
join completo - ogni tupla contribuisce ad almeno una tupla del risultato (vedi immagine sopra)

join con tuple dangling - alcune tuple non contribuiscono al risultato



join esterni - tutte le tuple contribuiscono al risultato, eventualmente estese con valori nulli

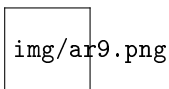
- sinistro
- destro
- completo



5.2.5 Prodotto cartesiano

Il risultato è la combinazione delle tuple in tutti i modi possibili.

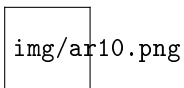
NB. il termine è improprio: il prodotto cartesiano di due insiemi è un insieme di coppie, qui abbiamo tuple.



5.2.6 Theta join e equi-join

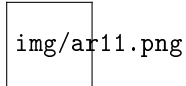
theta join - operatore derivato, prodotto cartesiano + selezione

equi-join - theta join in cui la condizione di selezione è una congiunzione di atomi di uguaglianza con un attributo della prima relazione



5.2.7 Logica a tre valori

vero (V), falso (F), sconosciuto (U)



5.2.8 Interrogazioni

Interrogazione: funzione che, applicata alla base di dati, produce relazioni.

5.3 Esercizi

5.3.1 Esercizio 1

Date le relazioni

STUDENTE(mat, nome, cognome, cittàNascita)

ESAME(matricola, codiceCorso, voto, data)

CORSO(codiceCorso, nomeCorso, docente, annoCreazione)

1. Trovare nome, cognome e matricola degli studenti che hanno superato l'esame di almeno un corso successivo alla data 10/10/2015
2. Trovare nome, cognome e matricola degli studenti che non hanno superato alcun esame (nota: solo gli esami con voto positivo vengono registrati nella relazione esame) dopo la data 10/10/2015
3. Trovare nome, cognome e matricola degli studenti omonimi (stesso nome e cognome) che sono nati nella stessa città

Abbiamo le seguenti tabelle:

STUDENTE			
<u>mat</u>	nome	cognome	cittàNascita

ESAME			
<u>matricola</u>	<u>codiceCorso</u>	voto	data

CORSO			
<u>codiceCorso</u>	nomeCorso	docente	annoCreazione

Ricordiamo come opera il join:

STUDENTE $\bowtie_{mat=matricola}$ ESAME						
mat	nome	cognome	cittàNascita	codiceCorso	voto	data

ESAME \bowtie CORSO						
matricola	codiceC.	voto	data	nomeC.	docente	annoC.

Soluzioni:

1. $\pi_{nome, cognome, mat}(\sigma_{annoCreazione > 2015}((STUDENTE \bowtie_{mat=matricola} ESAME) \bowtie CORSO))$
2. $\pi_{nome, cognome, matricola}(STUDENTE) - (\pi_{nome, cognome, matricola}(\sigma_{data > 10/10/2015}(STUDENTE \bowtie_{mat=matricola} ESAME)))$
3. $\pi_{nome, cognome, matricola}(\rho_{nome1, cognome1, citta1} \leftarrow_{nome, cognome, cittaNascita}(STUDENTE) \bowtie_{nome1=nome \wedge cognome1=cognome \wedge citta1=cittaNascita} STUDENTE)$

5.3.2 Esercizio 2

Date le relazioni:

STUDENTE(mat, nome, cognome)

ESAME(mat, codCorso, voto, data)

CORSO(codCorso, nomeCorso, docente)

1. Trovare nome, cognome e matricola degli studenti che hanno superato l'esame di almeno un corso
2. Trovare nome, cognome e matricola degli studenti che non hanno superato alcun esame (nota: solo gli esami con voto positivo vengono registrati nella relazione esame)
3. Trovare la matricola degli studenti che abbiano superato sia basi di dati che analisi (entrambi gli esami)

1. $\pi_{nome,cognome,mat}(\text{STUDENTE} \bowtie \text{ESAME})$
2. $\pi_{nome,cognome,mat}(\text{STUDENTE}) - \pi_{nome,cognome,mat}(\text{STUDENTE} \bowtie \text{ESAME})$
3. $\pi_{nome,cognome,mat}((\sigma_{corso.nomeCorso='Basi\ di\ dati'}(\text{STUDENTE} \bowtie \text{ESAME} \bowtie \text{CORSO})) \wedge (\sigma_{corso.nomeCorso='Analisi'}(\text{STUDENTE} \bowtie \text{ESAME} \bowtie \text{CORSO})))$

5.3.3 Esercizio 3

Si consideri il seguente schema relativo a una base di dati che raccoglie i turni degli assistenti di volo di una compagnia aerea. Il grado dell'assistente di volo può essere 'junior' o 'senior'. Per il turno, si suppone che un assistente di volo possa fare una sola tratta andata e ritorno al giorno (per esempio, nella stessa giornata può effettuare solo due viaggi sulla tratta Milano-Roma).

AssistenteDiVolo(CodiceA, Nome, Cognome, Genere, Grado, DataNascita)

Volo(CodiceVolo, Provenienza, Destinazione)

Turno(CodiceVolo, CodiceA, Data)

1. Trovare nome, cognome e data di nascita degli assistenti di volo di genere femminile con grado senior che abbiano effettuato il 22 giugno 2007 un volo Milano-Roma e un volo Roma-Atene.

$$\pi_{nome,cognome,dataNascita}(\sigma_{genere='F' \wedge grado='senior' \wedge data='22/06/2007'}(\text{ASSISTENTE} \bowtie \text{TURNO} \bowtie (\sigma_{provenienza='Milano' \wedge destinazione='Roma'}(\text{VOLO}) \cup \sigma_{provenienza='Roma' \wedge destinazione='Atene'}(\text{VOLO}))))$$

6 SQL

L'SQL (*Structured Query Language*) è un linguaggio standardizzato per database basati sul modello relazionale (RDBMS), progettato per le seguenti operazioni:

- creare e modificare schemi di database (**DDL** = Data Definition Language);
- inserire, modificare e gestire dati memorizzati (**DML** = Data Manipulation Language);
- interrogare i dati memorizzati (**DQL** = Data Query Language);
- creare e gestire strumenti di controllo e accesso ai dati (**DCL** = Data Control Language).

È dunque un linguaggio per interrogare e gestire basi di dati mediante l'utilizzo di costrutti di programmazione denominati **query**. La maggior parte delle implementazioni dispongono di interfaccia alla riga di comando per l'esecuzione diretta di comandi, in alternativa alla sola interfaccia grafica GUI.

6.1 Operatori

Gli operatori messi a disposizione dall'SQL standard si dividono in sette categorie:

- Operatori di assegnazione
- Operatori di confronto
- Operatori stringa
- Operatori aritmetici
- Operatori condizionali
- Operatori logici
- Operatori tra bit

6.1.1 Operatori di assegnazione

=	Esprime un'assegnazione e non restituisce alcun valore.
:=	Esprime un'assegnazione di un valore ad una variabile non ancora istanziata e non restituisce alcun valore.

6.1.2 Operatori di confronto

=	Esprime uguaglianza tra due valori numerici o stringhe di caratteri (dove non è usato come operatore di assegnazione)
IS	Si usa per verificare se un valore è NULL, oppure se corrisponde a un valore booleano (TRUE, FALSE, UNKNOWN).
LIKE	Esprime somiglianza tra due valori letterali. È possibile usare, per i confronti, i caratteri speciali % (sostituisce un arbitrario numero di lettere) e _ (sostituisce una lettera arbitraria)
<	Minore
>	Maggiore
<=	Minore o uguale
>=	Maggiore o uguale
<>	Diverso
!=	Diverso
BETWEEN ... AND	Recupera un valore compreso tra due valori
IN	Stabilisce se un valore è contenuto in una lista di valori possibili
EXISTS	Stabilisce se una determinata subquery restituisce un valore
ANY o SOME	Stabilisce se una determinata subquery restituisce almeno uno dei valori specificati
ALL	Stabilisce se una determinata subquery restituisce tutti i valori desiderati

Operatori contrari

IS NOT
NOT LIKE
NOT BETWEEN
NOT IN
NOT EXISTS

6.1.3 Operatori aritmetici

+	Addizione
-	Sottrazione
*	Moltiplicazione
/	Divisione
MOD	Restituisce il resto di una divisione
DIV	Restituisce la parte intera di una divisione

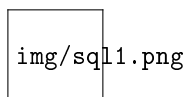
6.1.4 Operatori condizionali

L'unico operatore condizionale di SQL è WHERE (*dove*, vedi 6.2.1) e serve a definire criteri di ricerca mirati.

6.1.5 Operatori logici

AND	Restituisce il valore TRUE se e solo se entrambi gli operandi sono veri.
OR	Restituisce TRUE se e solo se almeno uno degli operandi è vero.
NOT	Restituisce falso se questo è vero, vero se questo è falso.
XOR	Restituisce TRUE se e solo se uno solo degli operandi è vero.

6.2 Interrogazioni



esempio di interrogazione

6.2.1 Select, from, where

clausola **select** (target list) - specifica gli elementi dello schema della tabella risultato. Come argomento può comparire *, che rappresenta tutti gli attributi delle tabelle elencate nella clausola from.

clausola **from** - ha come argomento l'insieme delle tabelle a cui si vuole accedere. Sul prodotto cartesiano delle tabelle vengono applicate le condizioni contenute nella clausola where.

clausola **where** - ha come argomento un'espressione booleana

Consideriamo le due tabelle

IMPIEGATO(Nome, Cognome, Dipart, Ufficio, Stipendio, Città)

DIPARTIMENTO(Nome, Indirizzo, Città)

1. estrarre lo stipendio degli impiegati di nome "Rossi".

```
select Stipendio as Salario
from Impiegato
where Cognome = 'Rossi'
```

2. estrarre tutte le informazioni relative agli impiegati di cognome "Rossi"

```
select *
from Impiegato
where Cognome = 'Rossi'
```

3. estrarre lo stipendio mensile dell'impiegato che ha cognome "Bianchi"

```
select Stipendio/12 as StipendioMensile
from Impiegato
where Cognome = 'Bianchi'
```

4. estrarre i nomi degli impiegati e le città in cui lavorano

```
select Impiegato.Nome, Impiegato.Cognome, Dipartimento.Città
from Impiegato, Dipartimento
where Impiegato.Dipart = Dipartimento.Nome
```

5. gli attributi per cui sorge un'ambiguità sono Nome e Città. L'interrogazione precedente può essere espressa facendo uso degli alias per le tabelle allo scopo di abbreviare i riferimenti a esse.

```
select I.Nome, Cognome, D.Città
from Impiegato as I, Dipartimento as D
where Dipart = D.Nome
```

6. estrarre nome e cognome degli impiegato che lavorano nell'ufficio 20 del dipartimento Amministrazione.

```
select Nome, Cognome
from Impiegato
where Ufficio = 20 and Dipart = 'Amministrazione'
```

7. estrarre i nomi e i cognomi degli impiegati che lavorano nel dipartimento Amministrazione o nel dipartimento Produzione

```
select Nome, Cognome
from Impiegato
where Dipart = 'Amministrazione' or
      Dipart = 'Produzione'
```

8. estrarre i nomi propri degli impiegati di nome "Rossi" che lavorano nei dipartimenti Amministrazione o Produzione

```
select Nome
from Impiegato
where Cognome = 'Rossi' and
      (Dipart = 'Amministrazione' or
       Dipart = 'Produzione')
```

6.2.2 Is null

predicato `is null` - per selezionare termini con valori nulli (\neg `is not null`)

6.2.3 Distinct, all

parola chiave `distinct` - per eliminare i duplicati (\neg `all`, di default, opzionale)

Consideriamo la relazione

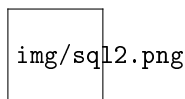
PERSONA(CodFiscale, Nome, Cognome, Città)

9. estrarre le città delle persone con cognome “Rossi”, facendo comparire ogni città al più una volta

```
select distinct Città
from Persona
where Cognome = 'Rossi'
```

6.2.4 Join, alias

operatore `join` - si scrive nell’ambito della clausola from, e non compare il where. (inner, right, left, full)



10. riscrivi il punto 5.

```
select I.Nome, Cognome, D.Città
from Impiegato I join Dipartimento D
on Dipart = D.Nome
```

11. estrarre tutti gli impiegati che hanno lo stesso cognome (ma diverso nome) di impiegati del dipartimento Produzione

```
select I1.Nome, I1.Cognome
from Impiegato I1, Impiegato I2
where I1.Cognome = I2.Cognome and
      I1.Nome <> I2.Nome and
      I2.Dipart = 'Produzione'
```

immaginiamo che al momento della definizione degli `alias` (I1 e I2) vengano create due diverse tabelle che verranno confrontate tra di loro.

6.2.5 Order by

clausola `order by` - ordina in modo ascendente (`asc`) o discendente (`desc`)

12. estrarre il contenuto della tabella AUTOMOBILE ordinato in base alla marca (in modo discendente) e al modello

```
select *
from Automobile
order by Marca desc, Modello
```

6.2.6 Operatori aggregati

`count` - torna il numero degli attributi
`sum` - torna la somma dei valori dell'espressione
`max` - massimo
`min` - minimo
`avg` - media dei valori

13. estrarre il numero di diversi valori dell'attributo Stipendio fra tutte le righe di IMPIEGATO

```
select count (distinct Stipendio)
from Impiegato
```

6.2.7 Group by

clausola `group by` - raggruppa per

14. estrarre la somma degli stipendi di tutti gli impiegati dello stesso dipartimento

```
select Dipart, sum(Stipendio)
from Impiegato
group by Dipartimento
```

6.2.8 Predicati su gruppi: having

predicato `having` - che ha

15. estrarre i dipartimenti che spendono più di 100 mila euro in stipendi

```
select Dipart, sum(Stipendio) as SommaStipendi
from Impiegato
group by Dipart
having sum(Stipendio) > 100
```

6.2.9 Interrogazioni di tipo insiemistico

`union`
`intersect`
`except`

16. estrarre i nomi e i cognomi degli impiegati

```
select Nome
from Impiegato
union
select Cognome
from Impiegato
```

6.2.10 Any

clausola `any` - la riga soddisfa la condizione se è vero il confronto tra il valore dell'attributo per la riga e almeno uno degli elementi restituiti dall'interrogazione

17. estrarre gli impiegati che lavorano in dipartimenti situati a Firenze

```
select *
from Impiegato
where Dipart = any (select Nome
                    from Dipartimento
                    where Città = 'Firenze')
```


6.2.11 Sintassi completa

```
SELECT [ DISTINCT ] lista_elementi_selezione
FROM lista_riferimenti_tabella
[ WHERE espressione_condizionale ]
[ GROUP BY lista_colonne ]
[ HAVING espressione_condizionale ]
[ ORDER BY lista_colonne ]
```

6.3 Viste



Mediante l'istruzione `CREATE VIEW` si definisce una vista, ovvero una "tabella virtuale". Le tuple della vista sono il risultato di una query che viene valutata dinamicamente ogni volta che si fa riferimento alla vista.

6.3.1 Esempio 1

A small rectangular box containing the text "img/sql3.png".

```
CREATE VIEW ProgSedi(CodProg, CodSede)
AS  SELECT P.CodProg, S.Sede
    FROM Prog P, Sedi S
    WHERE P.Città=S.Città;
```

```
SELECT *
FROM ProgSedi
WHERE CodProg = 'P01'
```

A small rectangular box containing the text "img/sql4.png".A small rectangular box containing the text "img/sql5.png".

6.3.2 Esempio 2

"Trova la sede che ha il massimo numero di impiegati"

```
CREATE VIEW NumImp(Sede, NImp)
AS SELECT Sede, COUNT(*)
   FROM Imp
   GROUP BY Sede;
```

```
SELECT Sede
FROM NumImp
WHERE NImp = (SELECT MAX(NImp)
              FROM NumImp)
```

6.4 Esercizi

6.4.1 Esercizio 1

«««< HEAD Data la seguente base di dati:

```
===== »»»> f674ee423c5a8cce825585ef2c25d72994b5de73 ATTORI(CodAttore,
Nome, AnnoNascita, Nazionalità)
RECITA(CodAttore*, CodFilm*)
FILM(CodFilm, Titolo, AnnoProduzione, Nazionalità, Regista, Genere)
PROIEZIONI(CodProiezione, CodFilm*, CodSala*, Incasso, DataProiezione)
«««< HEAD SALE(CodSala, Posti, Nome, Città)
```

Scrivere le interrogazioni SQL che restituiscono le seguenti informazioni:

1. Il nome di tutte le sale di Pisa

```
SELECT Nome FROM Sale
WHERE Città='Pisa'
```

2. Il titolo dei film di F. Fellini prodotti dopo il 1960.

```
SELECT Titolo FROM Film
WHERE Regista='Fellini' AND
      AnnoProduzione>1960
```

3. Il titolo e il regista dei film di fantascienza giapponesi o francesi prodotti dopo il 1990

```
SELECT Titolo, Regista FROM Film
WHERE Genere='fantascienza' AND
      (Nazionalità='giapponese' OR
       Nazionalità='francese') AND
      AnnoProduzione>1990
```

4. Il titolo dei film di fantascienza, giapponesi prodotti dopo il 1990 oppure francesi

```
SELECT Titolo FROM Film
WHERE Genere='fantascienza' AND
      ((Nazionalità='giapponese' AND
        AnnoProduzione>1990) OR
       Nazionalità='francese')
```

5. I titolo dei film dello stesso regista di “Casablanca”

```
SELECT Titolo FROM Film
WHERE Regista = (SELECT Regista FROM Film
                 WHERE Titolo='Casablanca')
```

6. Il titolo ed il genere dei film proiettati il giorno di Natale 2004

```
SELECT DISTINCT Titolo, Genere FROM Film
WHERE CodFilm = (SELECT CodFilm FROM Proiezioni
                 WHERE DataProiezione='25/12/2004')
```

7. Il titolo ed il genere dei film proiettati a Napoli il giorno di Natale 2004

```
SELECT DISTINCT Titolo, Genere FROM Film
WHERE CodFilm = (SELECT CodFilm FROM Proiezioni
                 WHERE DataProiezione='25/12/2004' AND
                 CodSala = (SELECT CodSala FROM Sale
                             WHERE Città='Napoli'))
```

oppure

```
SELECT DISTINCT Titolo, Genere FROM Film
JOIN Proiezioni ON Film.CodFilm=Proiezioni.CodFilm
JOIN Sale ON Proiezioni.CodSala=Sale.CodSala
WHERE Città='Napoli' AND
      DataProiezione='25/12/2004'
```

8. I nomi delle sale di Napoli in cui il giorno di Natale 2004 è stato proiettato un film con R.Williams

```
SELECT S.Nome FROM Sale S
JOIN Proiezioni P ON S.CodSala=P.CodSala
JOIN Recita R ON P.CodFilm=R.CodFilm
JOIN Attori A ON R.CodAttore=A.CodAttore
WHERE Città='Napoli' AND
      DataProiezione='25/12/2004' AND
      A.Nome='Williams'
```

oppure

```
SELECT Nome FROM Sale S
WHERE CodSala = (SELECT CodSala FROM Proiezioni
                 WHERE DataProiezione='25/12/2004' AND
                 CodFilm = (SELECT CodFilm FROM Recita
                             WHERE CodAttore = ( SELECT CodAttore
                                                    FROM Attori
                                                    WHERE Nome='Williams'))))
```

6.4.2 Esercizio 2

Data la seguente base di dati:

STUDENTE(mat, nome, cognome, luogoNascita)

ESAME(mat, codCorso, voto, data)

CORSO(codCorso, nomeCorso, docente)

Si scrivano le seguenti query SQL:

1. Trovare la città in cui sono nati gli studenti che complessivamente hanno superato più esami rispetto agli studenti nati in altre città.

```
CREATE VIEW Conteggio(città, numEsami)
AS SELECT S.luogoNascita, COUNT(S.mat)
FROM Studente S
JOIN Esame E ON S.mat=E.mat
GROUP BY S.luogoNascita;

SELECT C.città FROM Conteggio C
WHERE C.numEsami = (SELECT MAX(C2.numEsami)
                    FROM Conteggio C2)
```

2. Trovare nome, cognome e matricola degli studenti che hanno superato l'esame di analisi con voto superiore al voto ottenuto per l'esame di fisica.

```
SELECT S.nome, S.cognome, S.mat FROM Studente S
JOIN Esame E1 ON S.mat=E1.mat
JOIN Esame E2 ON S.mat=E2.mat
JOIN Corso C1 ON E1.codCorso=C1.codCorso
JOIN Corso C2 ON E2.codCorso=C2.codCorso
WHERE C1.nomeCorso='Analisi' AND
      C2.nomeCorso='Fisica' AND
      E1.voto>E2.voto
```

3. Trovare il nome dei corsi di cui studenti omonimi hanno superato l'esame.

```
SELECT C.nome FROM Corso C
JOIN ESAME E ON C.codCorso=E.codCorso
JOIN STUDENTE S1 ON E.mat=S1.mat
JOIN STUDENTE S2 ON E.mat=S2.mat
WHERE S1.nome=S2.nome AND
      S1.cognome=S2.cognome AND
      S1.mat<>S2.mat
```

6.4.3 Esercizio 3

La seguente base di dati descrive l'attività di una società di noleggio di autovetture che ha diverse sedi in alcune città italiane:

SEDE(nomeSede, Indirizzo, città, telefono, direttore)

AUTOVETTURA(targa, modello, annoImmatricolazione, numeroPosti, colore)

CLIENTE(CF, nome, cognome, telefono, città)

CONTRATTO(auto, cliente, dataNoleggio, sedeNoleggio, dataConsegna, sedeConsegna)

1. Trovare il numero di autovetture noleggiate in ogni città, da clienti con residenza in una città diversa dalla città in cui avviene il noleggio

```
SELECT C.sedeNoleggio, A.COUNT(targa) FROM Autovettura A
JOIN Contratto C ON A.targa = C.auto
JOIN Cliente C1 ON C.cliente = C1.CF
WHERE C1.città<>C.sedeNoleggio
GROUP BY C.sedeNoleggio
```

2. Trovare i clienti (CF, nome, cognome) che hanno noleggiato almeno due macchine diverse nella stessa sede di noleggio

```
SELECT DISTINCT C.CF, C.nome, C.cognome, FROM Cliente C
JOIN Contratto C1 ON C.CF=C1.cliente
JOIN Contratto C2 ON C.CF=C2.cliente
JOIN AUTOVETTURA A1 ON C1.auto=A1.targa
JOIN AUTOVETTURA A2 ON C2.auto=A2.targa
WHERE C1.sedeNoleggio=C2.sedeNoleggio AND
      A1.targa<>A2.targa
```

3. Trovare la sede di noleggio, indicandone anche la città, che ha realizzato il maggior numero di contratti della durata di un giorno (cioè in cui la macchina sia stata riconsegnata lo stesso giorno in cui è stata noleggiata)

#Raggruppa le righe in base alla sede e alla città
#e per ogni gruppo effettua il conteggio degli elementi presenti

```
CREATE VIEW NoleggiUnGiorno(sede, città, numero)
AS SELECT S.nomeSede, S.città, COUNT(*)
FROM Sede S
JOIN Contratto C ON S.nomeSede=C.sedeNoleggio
WHERE C.dataNoleggio=C.dataConsegna
GROUP BY S.nomeSede, S.città;

SELECT N.sede, N.città
FROM NoleggiUnGiorno N
WHERE numero = (SELECT MAX(N1.numero) FROM NoleggiUnGiorno N1)
```

6.4.4 Esercizio 4

La seguente basi di dati descrive l'archivio di una società di charter che organizza e vende vacanze in barca. La società dispone di una flotta di barche ed organizza viaggi con diverse destinazioni.

VIAGGIO(idCliente, dataPartenza, nomeBarca, Skipper, portoPartenza, portoArrivo, costo)

CLIENTE(idCliente, nome, cognome, nazione)

FLOTTA(NomeBarca, modello, lunghezza, posti, annoImmatricolazione)

PORTO(Nome, nazione, posti, telefono, canaleRadio)

1. Trovare la nazione che è stata coinvolta nella maggior parte dei viaggi (come porto di arrivo o come porto di partenza, entrambe)

```
CREATE VIEW Info as SELECT P.nazione, count(*) as Conteggio FROM Porto P
JOIN Viaggio V1 ON P.nazione = V.portoPartenza
JOIN Viaggio V2 ON P.nazione = V.portoArrivo
GROUP BY P.nazione;
```

```
SELECT I.nazione FROM Info I
WHERE I.Conteggio = SELECT(MAX(I1.Conteggio) FROM Info I1)
```

2. Trovare il nome delle barche che non sono mai state affittate

```
SELECT F.NomeBarca FROM Flotta F
LEFT JOIN Viaggio V ON F.NomeBarca=V.nomeBarca
WHERE V.nomeBarca IS NULL
```

3. Trovare per ogni porto il costo medio dei viaggi da lì partiti che non abbiamo imbarcato clienti della nazionalità del porto stesso.

```
SELECT P.Nome, AVG(V.costo)
FROM Porto P
JOIN Viaggio V on P.Nome=V.portoPartenza
JOIN Cliente C on P.nazione<>C.nazione
GROUP BY P.Nome
```

===== SALE(CodSala, Posti, Nome, Città)

Scrivere le interrogazioni SQL che restituiscono le seguenti informazioni

- (a) Il nome di tutte le sale di Pisa
- (b) Il titolo dei film di F. Fellini prodotti dopo il 1960.
- (c) Il titolo e il regista dei film di fantascienza giapponesi o francesi prodotti dopo il 1990
- (d) Il titolo dei film di fantascienza, giapponesi prodotti dopo il 1990 oppure francesi
- (e) I titolo dei film dello stesso regista di “Casablanca”
- (f) Il titolo ed il genere dei film proiettati il giorno di Natale 2004
- (g) Il titolo ed il genere dei film proiettati a Napoli il giorno di Natale 2004
- (h) I nomi delle sale di Napoli in cui il giorno di Natale 2004 è stato proiettato un film con R.Williams

Soluzioni

- (a) `SELECT Nome FROM Sale
WHERE Città='Pisa'`
- (b) `SELECT Titolo FROM Film
WHERE Regista='Fellini' AND
AnnoProduzione>1960`
- (c) `SELECT Titolo, Regista FROM Film
WHERE Genere='fantascienza' AND
(Nazionalità='giapponese' OR
Nazionalità='francese') AND
AnnoProduzione>1990`
- (d) `SELECT Titolo FROM Film
WHERE Genere='fantascienza' AND
((Nazionalità='giapponese' AND
AnnoProduzione>1990) OR
Nazionalità='francese')`
- (e) `SELECT Titolo FROM Film
WHERE Regista = (SELECT Regista FROM Film
WHERE Titolo='Casablanca')`
- (f) `SELECT DISTINCT Titolo, Genere FROM Film
WHERE CodFilm = (SELECT CodFilm FROM Proiezioni
WHERE DataProiezione='25/12/2004')`
- (g) `SELECT DISTINCT Titolo, Genere FROM Film
WHERE CodFilm = (SELECT CodFilm FROM Proiezioni
WHERE DataProiezione='25/12/2004' AND
CodSala = (SELECT CodSala FROM Sale
WHERE Città='Napoli'))`

oppure

```
SELECT DISTINCT Titolo, Genere FROM Film  
JOIN Proiezioni ON Film.CodFilm=Proiezioni.CodFilm  
JOIN Sale ON Proiezioni.CodSala=Sale.CodSala  
WHERE Città='Napoli' AND  
DataProiezione='25/12/2004'
```

- (h) `SELECT S.Nome FROM Sale S
JOIN Proiezioni P ON S.CodSala=P.CodSala
JOIN Recita R ON P.CodFilm=R.CodFilm
JOIN Attori A ON R.CodAttore=A.CodAttore
WHERE Città='Napoli' AND
DataProiezione='25/12/2004' AND
A.Nome='Williams'`

oppure

```
SELECT Nome FROM Sale S  
WHERE CodSala = (SELECT CodSala FROM Proiezioni  
WHERE DataProiezione='25/12/2004' AND  
CodFilm = (SELECT CodFilm FROM Recita  
WHERE CodAttore = (SELECT CodAttore FROM Attori  
WHERE Nome='Williams' )))
```

6.4.5 Esercizio 2

Data la seguente base di dati:

STUDENTE(mat, nome, cognome, luogoNascita)

ESAME(mat, codCorso, voto, data)

CORSO(codCorso, nomeCorso, docente)

Si scrivano le seguenti query SQL:

- (a) Trovare la città in cui sono nati gli studenti che complessivamente hanno superato più esami rispetto agli studenti nati in altre città.

```
SELECT TOP(1) luogoNascita FROM Studente S
JOIN Esame E ON S.mat=E.mat
GROUP BY luogoNascita
ORDER BY DISC COUNT(mat)
```

- (b) Trovare nome, cognome e matricola degli studenti che hanno superato l'esame di analisi con voto superiore al voto ottenuto per l'esame di fisica.

```
SELECT S.nome, S.cognome, S.mat FROM Studente S
JOIN Esame E1 ON S.mat=E1.mat
JOIN Esame E2 ON S.mat=E2.mat
JOIN Corso C1 ON E1.codCorso=C1.codCorso
JOIN Corso C2 ON E2.codCorso=C2.codCorso
WHERE C1.nomeCorso='Analisi' AND
      C2.nomeCorso='Fisica' AND
      E1.voto>E2.voto
```

- (c) Trovare il nome dei corsi di cui studenti omonimi hanno superato l'esame.

```
SELECT C.nome FROM Corso C
JOIN ESAME E ON C.codCorso=E.codCorso
JOIN STUDENTE S1 ON E.mat=S1.mat
JOIN STUDENTE S2 ON E.mat=S2.mat
WHERE S1.nome=S2.nome AND
      S1.cognome=S2.cognome AND
      S1.mat<>S2.mat
```

6.4.6 Esercizio 3

La seguente base di dati descrive l'attività di una società di noleggio di autovetture che ha diverse sedi in alcune città italiane:

SEDE(nomeSede, Indirizzo, città, telefono, direttore)

AUTOVETTURA(targa, modello, annoImmatricolazione, numeroPosti, colore)

CLIENTE(CF, nome, cognome, telefono, città)

CONTRATTO(auto, cliente, dataNoleggio, sedeNoleggio, dataConsegna,
sedeConsegna)

- (a) Trovare il numero di autovetture noleggiate in ogni città, da clienti con residenza in una città diversa dalla città in cui avviene il noleggio

```
SELECT C.sedeNoleggio, A.COUNT(targa) FROM Autovettura A
JOIN Contratto C ON A.targa = C.auto
JOIN Cliente Cl ON C.cliente = Cl.CF
WHERE Cl.città<>C.sedeNoleggio
GROUP BY C.sedeNoleggio
```

- (b) Trovare i clienti (CF, nome, cognome) che hanno noleggiato almeno due macchine diverse nella stessa sede di noleggio

```
SELECT DISTINCT C.CF, C.nome, C.cognome, FROM Cliente C
JOIN Contratto C1 ON C.CF=C1.cliente
JOIN Contratto C2 ON C.CF=C2.cliente
JOIN AUTOVETTURA A1 ON C1.auto=A1.targa
JOIN AUTOVETTURA A2 ON C2.auto=A2.targa
WHERE C1.sedeNoleggio=C2.sedeNoleggio AND
A1.targa<>A2.targa
```

- (c) Trovare la sede di noleggio, indicandone anche la città, che ha realizzato il maggior numero di contratti della durata di un giorno (cioè in cui la macchina sia stata riconsegnata lo stesso giorno in cui è stata noleggiata)

```
#Raggruppa le righe in base alla sede e alla città
#e per ogni gruppo effettua il conteggio degli elementi presenti
```

```
CREATE VIEW NoleggiUnGiorno(sede, città, numero)
AS SELECT S.nomeSede, S.città, COUNT(*)
FROM Sede S
JOIN Contratto C ON S.nomeSede=C.sedeNoleggio
WHERE C.dataNoleggio=C.dataConsegna
GROUP BY S.nomeSede, S.città;

SELECT N.sede, N.città
FROM NoleggiUnGiorno N
WHERE numero = (SELECT MAX(N1.numero) FROM NoleggiUnGiorno N1)
```

6.4.7 Esercizio 4

La seguente basi di dati descrive l'archivio di una società di charter che organizza e vende vacanze in barca. La società dispone di una flotta di barche ed organizza viaggi con diverse destinazioni.

VIAGGIO(idCliente, dataPartenza, nomeBarca, Skipper, portoPartenza, portoArrivo, costo)

CLIENTE(idCliente, nome, cognome, nazione)
 FLOTTA(NomeBarca, modello, lunghezza, posti, annoImmatricolazione)
 PORTO(Nome, nazione, posti, telefono, canaleRadio)

- (a) Trovare la nazione che è stata coinvolta nella maggior parte dei viaggi (come porto di arrivo o come porto di partenza, entrambe)

```

CREATE VIEW Info as SELECT P.nazione, count(*) as Conteggio FROM Porto P
JOIN Viaggio V1 ON P.nazione = V.portoPartenza
JOIN Viaggio V2 ON P.nazione = V.portoArrivo
GROUP BY P.nazione;

```

```

SELECT I.nazione FROM Info I
WHERE I.Conteggio = SELECT(MAX(I1.Conteggio) FROM Info I1)

```

- (b) Trovare il nome delle barche che non sono mai state affittate

```

SELECT F.NomeBarca FROM Flotta F
LEFT JOIN Viaggio V ON F.NomeBarca=V.nomeBarca
WHERE V.nomeBarca IS NULL

```

- (c) Trovare per ogni porto il costo medio dei viaggi da lì partiti che non abbiamo imbarcato clienti della nazionalità del porto stesso.

```

SELECT P.Nome, AVG(V.costo)
FROM Porto P
JOIN Viaggio V on P.Nome=V.portoPartenza
JOIN Cliente C on P.nazione<>C.nazione
GROUP BY P.Nome

```

»»»> f674ee423c5a8cce825585ef2c25d72994b5de73

6.4.8 Esercizio 5

Data la seguente base di dati:

AGENTE(matricola, nome, cognome, telefono, annoNascita, annoIscrizioneAlbo)
 LOCALE(idLocale, tipoLocale, indirizzo, città, mq, codiceProprietario)
 AFFITTO(idLocale, agente, dataStipulaContratto, codiceAffittuario, dataFineContratto, costoMensile)
 VENDITA(idLocale, agente, prezzo, dataVendita, codiceAcquirente)
 CLIENTE(codiceCliente, nome, cognome, telefono, CF)

«««< HEADTrovare codiceCliente, nome, cognome dei clienti che abbiano sia venduto che affittato locali a Milano nel 2010

- (a) SELECT C.codiceCliente, C.nome, C.cognome FROM Cliente C
 JOIN Locale L ON C.codiceCliente=L.codiceProprietario
 JOIN Affitto A ON L.idLocale=A.idLocale
 JOIN Vendita V ON L.idLocale=V.idLocale
 WHERE L.città='Milano' AND
 YEAR(A.dataStipulaContratto)='2010' AND
 YEAR(V.dataVendita)='2010'

- (b) Trovare matricola, nome, cognome dell'agente che ha stipulato più contratti di vendita di uffici

```
#Conteggio è la nuova tabella
#Contratti è il nome della colonna che contiene count(*)
#Group by va fatto per forza su tutti gli elementi
#presenti nel SELECT che non sono generati da un operatore
#(come count() ad es.)
```

```
CREATE VIEW Conteggio as
SELECT A.matricola, A.nome, A.cognome, count(*) as Contratti
FROM Agente A
JOIN Vendita V ON A.matricola=V.agente
JOIN Loacle L ON V.idLocale=L.idLocale
WHERE tipoLocale='ufficio'
GROUP BY matricola, nome, cognome;

SELECT C.matricola, C.nome, C.cognome FROM Conteggio C
WHERE C.Contratti = ( SELECT MAX(C1.Contratti)
                     FROM Contratti C1)
```

=====

- (c) Trovare codiceCliente, nome, cognome dei clienti che abbiano sia venduto che affittato locali a Milano nel 2010

```
SELECT C.codiceCliente, C.nome, C.cognome FROM Cliente C
JOIN Locale L ON C.codiceCliente=L.codiceProprietario
JOIN Affitto A ON L.idLocale=A.idLocale
JOIN Vendita V ON L.idLocale=V.idLocale
WHERE L.città='Milano' AND
      YEAR(A.dataStipulaContratto)='2010' AND
      YEAR(V.dataVendita)='2010'
```

- (d) Trovare matricola, nome, cognome dell'agente che ha stipulato più contratti di vendita di uffici

```
#Conteggio è la nuova tabella
#Contratti è il nome della colonna che contiene count(*)
#Group by va fatto per forza su tutti gli elementi
#presenti nel SELECT che non sono generati da un operatore
#(come count() ad es.)
```

```
CREATE VIEW Conteggio as
SELECT A.matricola, A.nome, A.cognome, count(*) as Contratti
FROM Agente A
JOIN Vendita V ON A.matricola=V.agente
JOIN Loacle L ON V.idLocale=L.idLocale
WHERE tipoLocale='ufficio'
GROUP BY matricola, nome, cognome;

SELECT C.matricola, C.nome, C.cognome FROM Conteggio C
```

```
WHERE C.Contratti = (SELECT MAX(C1.Contratti) FROM Contratti C1)
```

```
»»»> f674ee423c5a8cce825585ef2c25d72994b5de73
```

7 Link utili

- <https://www.w3schools.com/sql/>
- <https://www.html.it/guide/guida-linguaggio-sql/>
- http://www.dia.uniroma3.it/~patrigna/didactic/sistemi_informativi/materiale.html
- <http://www.di.unito.it/~anselma/lingue/#testi>
- <http://www.di.unipi.it/~leoni/BDeSI/BD.html>
- <https://users.dimi.uniud.it/~massimo.franceschet/teatro-sql/>
- <http://www.star.dist.unige.it/~marco/SI1-SV/materialeDidattico.htm>
- <http://www.cs.unibo.it/~moretti/teaching.html>
- <http://www.dii.unisi.it/~franco/Teaching/BasiDiDati/0910/BasiDiDati.php>
- http://linuxdidattica.org/docs/fb_db/