

Accesso ai Dati nei Sistemi Relazionali

I Sistemi Relazionali offrono in generale due modalità di uso della Base di Dati

- ❖ Uso **interattivo**
- ❖ Uso **da programmi**

Uso interattivo:

l'utente presenta al sistema una *richiesta* di dati. Tale richiesta prende il nome di **interrogazione** (*query*).

L'interrogazione viene *interpretata* dal sistema, che in risposta restituisce i dati richiesti.

Nella richiesta devono essere specificate le proprietà dei dati che interessano. Se ad es. vogliamo l'elenco dei libri scritti da Calvino, nella richiesta deve essere specificata questa proprietà.

L'interrogazione deve essere formulata per mezzo di un linguaggio formale

Uso da programmi:

questo uso è riservato ad utenti programmatori. Le interrogazioni fanno parte di un programma applicativo che può essere eseguito dal sistema numerose volte, ed il risultato delle interrogazioni può essere utilizzato dal programma per successive elaborazioni

Ci limiteremo a trattare l'uso interattivo della BD.

SQL per definire Interrogazioni sulla Base di Dati

Le interrogazioni devono essere scritte in un linguaggio formale con caratteristiche tali da renderlo adatto ad esprimere interrogazioni sulla BD, e da essere facilmente interpretato dal sistema.

Il linguaggio generalmente usato si chiama **SQL** (Structured Query Language).

ESEMPIO Si consideri il seguente schema relazionale

Catalogo (ISBN, Titolo, CasaEd, AnnoEd)

Supponiamo che interessi conoscere il titolo e la la casa editrice dei libri pubblicati nel 2001. Occorre:

- consultare la relazione **Catalogo**

ISBN	Titolo	CasaEd	AnnoEd
883860789-9	Baudolino	Mondadori	2002
343859804-5	La storia	Einaudi	2001
110786949-3	Bar sport	Feltrinelli	2000
477800355-2	L'amante	Mondadori	2001

- considerare solo le ennuple in cui **AnnoEd = 2001**

ISBN	Titolo	CasaEd	AnnoEd
343859804-5	La storia	Einaudi	2001
477800355-2	L'amante	Mondadori	2001

- prelevare da queste ennuple i valori degli attributi **Titolo** e **CasaEd**

Titolo	CasaEd
La storia	Einaudi
L'amante	Mondadori

Questa sequenza di operazioni viene eseguita dal Sistema di Gestione di Basi di Dati, purché gli venga trasmesso un opportuno comando (interrogazione) nel linguaggio SQL:

```
SELECT Titolo, CasaEd  
FROM Catalogo  
WHERE AnnoEd = 2001
```

ove SELECT, FROM e WHERE sono parole riservate del linguaggio SQL.

Una **interrogazione** SQL agisce sulle relazioni definite nella base di dati, e **restituisce come risultato una relazione**.

Questa viene in generale visualizzata sul monitor, oppure stampata; può anche essere memorizzata nella base di dati o può essere utilizzata in altre interrogazioni.

Nei casi più semplici una interrogazione SQL deve specificare

- Quali sono le informazioni che interessano
- In quali relazioni si trovano
- Quali proprietà devono avere

Quali sono le informazioni che interessano

SELECT *Attributo₁, Attributo₂,...*

è presente in ogni interrogazione e definisce lo schema della relazione risultato. Più avanti vedremo che può avere una forma più complessa.

SELECT *Titolo, CasaEd*

significa che ci interessano il titolo e la casa editrice

In quali relazioni si trovano

FROM *Relazione₁, Relazione₂,...*

è presente in ogni interrogazione e specifica quali relazioni occorre visitare per ottenere il risultato.

FROM *Catalogo*

significa che per estrarre le informazioni che interessano occorre prendere in esame la relazione Catalogo

Quali proprietà devono essere soddisfatte

WHERE Condizione

La condizione è espressa sugli attributi delle relazioni specificate nella clausola FROM.
Può non essere presente, quando non si vogliono specificare condizioni.

WHERE AnnoEd = 2001

significa che interessano informazioni relative ai libri editi nel 2001.

Abbiamo visto che l'interrogazione

```
SELECT Titolo, CasaEd  
FROM Catalogo  
WHERE AnnoEd = 2001
```

restituisce la relazione

Titolo	CasaEd
La storia	Einaudi
L'amante	Mondadori

contenente titolo e casa editrice dei libri editi nel 2001

Invece l'interrogazione

```
SELECT Titolo, CasaEd  
FROM Catalogo
```

restituisce la relazione

Titolo	CasaEd
Baudolino	Mondadori
La storia	Einaudi
Bar sport	Feltrinelli
L'amante	Mondadori

contenente titolo e casa editrice di **tutti** i libri presenti nel catalogo

Riepilogando la forma generale di un'interrogazione SQL è, nei casi più semplici, la seguente:

```
SELECT Attributo1, Attributo2, ...  
FROM Relazione1, Relazione2, ...  
WHERE Condizione
```

- Le parole in maiuscolo sono parole riservate del linguaggio SQL, sono fisse e specificano le *clausole* dell'interrogazione; la clausola WHERE può mancare
- Le parole in minuscolo sono variabili, e rappresentano le relazioni, gli attributi, le condizioni che riguardano la specifica interrogazione

5. Linguaggio per Interrogazioni SQL

Si consideri la seguente interrogazione

```
SELECT CasaEd  
FROM Catalogo
```

Il risultato sarà

CasaEd
Mondadori
Einaudi
Feltrinelli
Mondadori

e se una casa editrice è presente nel catalogo con 1000 libri, il suo nome comparirà 1000 volte nel risultato

Se vogliamo evitare che ciò avvenga, scriveremo

```
SELECT DISTINCT CasaEd  
FROM Catalogo
```

Che ha come risultato le case editrici presenti nel catalogo, rappresentate una sola volta

CasaEd
Mondadori
Einaudi
Feltrinelli

In generale la specifica **DISTINCT** nella clausola **SELECT** elimina i duplicati dal risultato

La forma generale di un interrogazione SQL che abbiamo visto fin qui è quindi la seguente:

```
SELECT [DISTINCT] Attributo1, Attributo2,...  
FROM Relazione1, Relazione2,...  
[ WHERE Condizione ]
```

dove le parti racchiuse tra parentesi quadre possono mancare

L'uso di * - Nella clausola **SELECT** si può specificare ***** in luogo della lista di attributi; in tal caso il risultato contiene tutti gli attributi delle relazioni specificate nella clausola **FROM**

```
SELECT *  
FROM Catalogo  
WHERE CasaEd = "Feltrinelli"
```

Restituisce come risultato

ISBN	Titolo	CasaEd	AnnoEd
110786949-3	Bar sport	Feltrinelli	2000

WHERE Condizione

Fin'ora abbiamo considerato esempi molto semplici, ma la condizione presente nella clausola WHERE può avere una struttura molto complessa

In generale le condizioni sono formate combinando **predicati** con gli operatori booleani **and**, **or** e **not**.

- AnnoEd > 1980 **and** CasaEd = "Feltrinelli"
- AnnoEd = 2000 **and** (CasaEd = "Einaudi" **or** CasaEd = "Mondadori")

Predicato: è una condizione semplice del tipo

$$E_1 \text{ cfr } E_2$$

ove:

- **cfr** è un operatore di confronto, cioè uno degli operatori
= | < | > | <= | >= | <> (diverso)
- E_1 ed E_2 sono espressioni, che possono essere attributi, costanti oppure espressioni formate con gli usuali operatori aritmetici. Molto spesso E_1 è un attributo. E_2 può essere un comando SELECT

I predicati hanno valore *true* (vero) oppure *false* (falso).

Essi possono essere combinati tra loro con gli *operatori booleani*

and | or | not

Gli operatori booleani rispettano le seguenti *tabelle di verità*:

true and true = true	true or true = true	not true = false
true and false = false	true or false = true	not false = true
false and false = false	false or false = false	

La condizione presente nella clausola **WHERE** è ottenuta combinando predicati con gli operatori booleani; anch'essa quindi assume valore *true* oppure *false*, in accordo con le tabelle di verità.

Gli attributi che compaiono nei predicati devono appartenere alle relazioni presenti nella clausola **FROM**.

Più avanti vedremo altre modalità per esprimere le condizioni.

ESEMPI Consideriamo il solito schema di relazione **Catalogo** (ISBN, Titolo, CasaEd, AnnoEd) e una sua istanza

Catalogo

ISBN	Titolo	CasaEd	Anno
883860789-9	Baudolino	Mondadori	2002
446830226-7	L'amante	Feltrinelli	2001
343859804-5	La storia	Einaudi	2001
110786949-3	Bar sport	Feltrinelli	2000
477800355-2	L'amante	Mondadori	2001

```
SELECT Titolo, CasaEd  
FROM Catalogo  
WHERE Anno = 2001 and CasaEd = "Einaudi"
```

risultato

Titolo	CasaEd
La storia	Einaudi

5. Linguaggio per Interrogazioni SQL

```
SELECT Titolo, CasaEd
FROM Catalogo
WHERE Anno = 2001 or CasaEd = "Einaudi"
```

risultato

Titolo	CasaEd
L'amante	Feltrinelli
La storia	Einaudi
L'amante	Mondadori

```
SELECT Titolo, CasaEd
FROM Catalogo
WHERE Anno = 2000 and CasaEd <> "Feltrinelli"
```

Risultato

Titolo	CasaEd
--------	--------

Il risultato di un'interrogazione, che è sempre una relazione, può essere la relazione vuota; ciò significa, come nell'ultimo esempio, che non vi sono dati che soddisfano alla condizione.

```
SELECT CasaEd, Anno
FROM Catalogo
WHERE Titolo = "Bibbia" and
        Anno = (SELECT max(Anno)
                FROM Catalogo
                WHERE Titolo = "Bibbia" )
```

Viene dapprima calcolata la **SELECT** tra parentesi, ed il suo risultato viene utilizzato per valutare la condizione; La **SELECT** esterna restituisce come risultato la CasaEd e L'Anno della più recente edizione della Bibbia presente nel Catalogo (naturalmente il Catalogo non è quello dell'esempio)

Nell'ultimo esempio è stato fatto uso di una struttura detta *SOTTOSELECT*, o *SELECT annidata*.

Questa ha lo scopo di estrarre dalla BD un valore da utilizzare in una espressione

Si osservi che la Sottoselect ha come risultato un singolo valore, altrimenti il confronto non si può effettuare

Più avanti vedremo altri predicati per i quali non sussiste più questo vincoli, cioè la Sottoselect può avere come risultato una qualunque relazione.

Interrogazioni su più relazioni

Nella clausola FROM possono essere presenti più relazioni. Ciò è necessario quando le informazioni per eseguire l'interrogazione sono distribuite su relazioni diverse, vale a dire:

quando gli attributi presenti nella clausola SELECT o nella clausola WHERE appartengono a relazioni diverse

ESEMPIO Si consideri il seguente schema relazionale

Film(CodFilm, Titolo, Regista, Anno)
Attori(CodFilm^{*}, Attore)

e supponiamo di volere i titoli dei film in cui recita C. Eastwood

L'attributo Titolo è nella relazione Film mentre l'attributo Attore è nella relazione Attori. Occorre pertanto visitare entrambe le relazioni.

Supponiamo di avere le seguenti istanze di relazione:

FILM

<u>CodFilm</u>	<u>Titolo</u>	<u>Regista</u>	<u>Anno</u>
PW54	Million dollar baby	C.Eastwood	2004
MX23	Per un pugno di dollari	S.Leone	1964
AY78	Eyes wide shut	S.Kubrik	1999

ATTORI

<u>CodFilm</u>	<u>Attore</u>
PW54	C. Eastwood
MX23	C. Eastwood
PW54	H. Swank
PW54	M. Freeman
MX23	G. M. Volonté
AY78	T. Cruise
AY78	N. Kidman

Il SGBD esegue la seguente procedura:

- 1_ Viene costruita una relazione concatenando le ennuple di Film e di Attori che sono in associazione (tali che CodFilm=CodFilm*)

FILM&ATTORI

<u>CodF</u>	<u>Titolo</u>	<u>Regista</u>	<u>Ann</u>		<u>Attore</u>
PW54	Million dollar baby	C.Eastwood	2004	PW54	C.Eastwood
MX23	Per un pugno di dollari	S.Leone	1964	MX23	C.Eastwood
MX23	Per un pugno di dollari	S.Leone	1964	MX23	G.M.Volonté
PW54	Million dollar baby	C.Eastwood	2004	PW54	H. Swank
PW54	Million dollar baby	C.Eastwood	2004	PW54	M.Freeman
AY78	Eyes wide shut	S.Kubrik	1999	AY78	T.Cruise
AY78	Eyes wide shut	S.Kubrik	1999	AY78	N.Kidman

2_ Vengono prese in considerazione solo le ennuple in cui l'attributo Attore ha valore C.Eastwood.

FILM con C.Eastwood

CodF	Titolo	Regista	Ann		Attore
PW54	Million dollar baby	C.Eastwood	2004	PW54	C.Eastwood
MX23	Per un pugno di dollari	S.Leone	1964	MX23	C.Eastwood

3_ Viene prelevato l'attributo Titolo

Titolo dei FILM con C.Eastwood

Titolo
Million dollar baby
Per un pugno di dollari

Questa interrogazione in SQL si scrive:

```
SELECT Titolo
FROM Film, Attori
WHERE Film.CodFilm = Attori.CodFilm
and Attore = "C.Eastwood"
```

dove Film.CodFilm ed Attori.CodFilm rappresentano il valore di CodFilm nella relazione Film e nella relazione Attori rispettivamente.

La condizione **Film.CodFilm = Attori.CodFilm** serve ad esprimere il collegamento tra le ennuple di Film e quelle di Attori. Solo in questo modo C.Eastwood sarà associato ad un film in cui recita.

Sottolineiamo il fatto che fra le due relazioni deve esistere un collegamento (una chiave esterna in una relazione, chiave primaria nell'altra), e che nella clausola WHERE dell'interrogazione deve essere esplicitato tale collegamento.

Osserviamo che nella clausola FROM può essere presente un qualunque numero di relazioni, purché queste siano collegate tra di loro, e nella clausola WHERE siano specificati tutti i collegamenti.

GIUNZIONE

L'operazione che associa le ennuple di due relazioni (ad es. le ennuple di Film con quelle di Attori) è detta **giunzione**, e le condizioni di eguaglianza tra la chiave esterna di una relazione e la chiave primaria di un'altra (ad es. $\text{Film.CodFilm} = \text{Attori.CodFilm}$) è detto *predicato di giunzione*.

Precisazione sulla notazione

Per motivi di chiarezza e per evitare ambiguità, è opportuno specificare, per ogni attributo, la relazione cui appartiene, con la notazione *Relazione.Attributo*. Pertanto l'interrogazione precedente diventa

```
SELECT Film.Titolo
FROM Film, Attori
WHERE Film.CodFilm = Attori.CodFilm
and Attori.Attore = "C.Eastwood"
```

Per motivi di brevità è opportuno assegnare nella clausola FROM un nome abbreviato alle relazioni, da utilizzare nelle altre clausole dell'interrogazione:

```
SELECT F.Titolo
FROM Film F, Attori A
WHERE F.CodFilm = A.CodFilm
and A.Attore = "C.Eastwood"
```

Una interrogazione può essere complessa anche se la richiesta è molto semplice.
Parliamo naturalmente di complessità dal punto di vista dell'utente: la complessità per il calcolatore è un'altra cosa.

ESEMPIO

Si consideri lo schema relazionale:

FILM(CodiceVHS, Titolo, Regista, Anno)
ATTORI(Nome, Nazionalità)
RECITA(CodiceVHS*, Nome*, Personaggio)
CASSETTE(Collocazione, CodiceVHS*, DataNoleg,
CodiceCliente*)
CLIENTI(CodiceCliente, Cognome, Nome,
Indirizzo, Telefono)

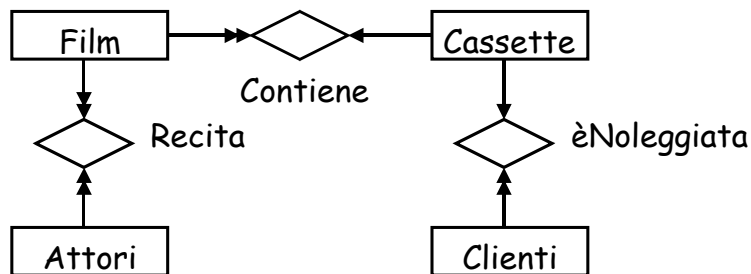
e si voglia estrarre Cognome e Nome dei Clienti che hanno noleggiato cassette relative a film in cui recitano attori francesi

Cognome e Nome sono attributi della relazione Clienti. Questa è collegata a Cassette tramite CodiceCliente, Cassette è collegata a Film tramite CodiceVHS, Film è collegato a Recita tramite CodiceVHS, ed infine Recita è collegato con Attori tramite Nome; finalmente in Attori troviamo l'attributo Nazionalità, e possiamo quindi verificare la condizione di ricerca.

In SQL tale interrogazione è piuttosto fastidiosa da scrivere:

```
SELECT Cl.Cognome, Cl.Nome
FROM  Clienti Cl, Cassette. Ca, Film F, Recita R,
      Attori A
WHERE Cl.CodiceCliente = Ca.CodiceCliente
      and Ca.CodiceVHS = F.CodiceVHS
      and F.CodiceVHS = R.CodiceVHS
      and R.Nome = A.Nome
      and A.Nazionalità = "francese"
```

Può essere utile, per individuare le relazioni da specificare nella clausola FROM, considerare lo schema E-R rappresentato dallo schema relazionale: da tale schema risulta evidente che per collegare Clienti con Attori occorre *attraversare* tutte le classi intermedie.



R.Gori - G.Leoni

Il linguaggio per interrogazioni SQL

43

La clausola **ORDER BY**

La clausola ORDER BY, specificata dopo SELECT-FROM-WHERE fa sì che il risultato sia ordinato; si può scegliere fra ordinamento crescente (se non si specifica nulla), o decrescente (se si specifica **desc**).

L'ordinamento può essere fatto anche su più attributi.

```
SELECT Cl.Cognome, Cl.Nome, Ca.DataNoleg
FROM Clienti Cl, Cassette Ca
WHERE Cl.CodiceCliente = Ca.CodiceCliente
      and Ca.DataNoleg < 31/03/05
ORDER BY Cl.Cognome, Cl.Nome
```

R.Gori - G.Leoni

Il linguaggio per interrogazioni SQL

44

Funzioni di aggregazione

SQL consente di estrarre dalla Base di Dati informazioni che non sono esplicitamente presenti, ma si ottengono da quelle presenti utilizzando opportune funzioni dette *funzioni di aggregazione*.

ESEMPIO

Studenti (Matricola, Nome, CorsodiLaurea)
Esami (Matricola*, CodiceAF*, Voto)
AttivitàFormativa(CodiceAF, NomeAF, CFU)

Le funzioni di aggregazione consentono di estrarre dalla BD informazioni quali il numero di esami sostenuti da un determinato studente, il numero di studenti che hanno sostenuto un determinato esame, valori medi, massimi, minimi ecc.

? Numero di esami sostenuti dallo studente con Matricola 123

```
SELECT Count(*)  
FROM Esami  
WHERE Matricola=123
```

Count(*) indica un conteggio: vengono contate le ennuple (ricordiamo che * indica l'intera ennupla) di Esami che soddisfano alla condizione Matricola=123.

? Il voto più basso dello studente con Matricola 123

```
SELECT Min(Voto)
FROM Esami
WHERE Matricola=123
```

? Numero di studenti che hanno sostenuto l'esame con CodiceAF AA252

```
SELECT Count(*)
FROM Esami E
WHERE CodiceAF=AA252
```

Al risultato di una funzione di aggregazione può essere dato un nome tramite il costrutto **as**:

```
SELECT Count(*) as Numero_Esami_AA252
FROM Esami E
WHERE CodiceMateria=AA252
```

? Numero di crediti acquisiti dallo studente con Matricola 123

```
SELECT Sum(CFU) as Crediti_di_123
FROM Esami E, AttivitàFormative A
WHERE E.CodiceAF = A.CodiceAF
and E.Matricola=123
```

Sum(CFU) indica l'ordinaria somma aritmetica dei valori (che devono essere numerici) dell'attributo CFU

Attenzione!

La funzione Sum non va confusa con la funzione Count

Per comprendere quest'ultima interrogazione occorre ricordare che con la giunzione specificata dalla condizione $E.CodiceAF = A.CodiceAF$ vengono concatenate le ennuple di Esami con quelle "corrispondenti" di AttivitàFormative e poi vengono estratte quelle che soddisfano alla condizione $E.Matricola = 123$. Con la funzione Sum viene poi eseguita la somma dei valori dell'attributo CFU di tali ennuple.

Consideriamo a seguente istanza della BD

5. Linguaggio per Interrogazioni SQL

Attività Formative

CodAF	NomeAF	CFU
AA252	Basi di..	5
BB112	Storia...	10
CC205	Letter..	10

Esami

Matric	CodAF	Voto
123	AA252	30
234	BB112	24
345	AA252	28
123	BB112	27
123	CC205	22
345	BB112	27

La giunzione $E.CodiceAF = A.CodiceAF$ da luogo alla seguente relazione

Attività Formative&Esami

CodAF	NomeAF	CFU	Matric	CodAF	Voto
AA252	Basi di..	5	123	AA252	30
AA252	Basi di..	5	345	AA252	28
BB112	Storia...	5	234	BB112	24
BB112	Storia...	10	123	BB112	27
BB112	Storia...	10	345	BB112	27
CC205	Letter..	10	123	CC205	22

La condizione $E.Matricola=123$ da luogo alla seguente relazione

Attività Formative & Esami di 123

CodAF	NomeAF	CFU	Matric	CodAF	Voto
AA252	Basi di..	5	123	AA252	30
BB112	Storia...	10	123	BB112	27
CC205	Letter..	10	123	CC205	22

La funzione Sum esegue la somma dei valori dell'attributo CFU e si ottiene il risultato desiderato

Crediti_di_123

25

L'uso delle funzioni di aggregazione è limitato al caso in cui il risultato sia costituito da un solo valore; non possono cioè essere presenti allo stesso tempo nella clausola SELECT sia attributi che funzioni di aggregazione.

La seguente interrogazione, ad es. è sbagliata (del resto avrebbe poco senso)

```
SELECT Voto, Count(*)
FROM Esami
WHERE Matricola=123
```

Vedremo tra breve un uso più interessante delle funzioni di aggregazione.

Le funzioni di aggregazione previste da SQL sono:

avg media aritmetica (valori numerici)

count numero di valori

max valore massimo

min valore minimo

sum somma (valori numerici)

Min e Max, quando sono applicati a valori non numerici, danno rispettivamente il primo e l'ultimo valore nell'ordine alfabetico.

La clausola **GROUP BY**

Si presenta sovente la necessità di avere informazioni su dati aggregati in base a qualche criterio. Ad es. il numero di esami sostenuti da ciascuno studente, il numero di studenti iscritti a ciascun Corso di Laurea, ...

SQL offre la possibilità di estrarre tali informazioni con l'uso dell'operatore **GROUP BY**.

? Numero di studenti iscritti a ciascun Corso di Laurea

```
SELECT CorsodiLaurea, count(*) as N°Studenti  
FROM Studenti  
GROUP BY CorsodiLaurea
```

Le ennuple di Studenti vengono raggruppate in base al Valore dell'attributo CorsodiLaurea; ciascun gruppo genera una ennupla del risultato:

Corso di Laurea, Numero di Studenti del CdL

Osserviamo che nella clausola Select son presenti sia funzioni di aggregazione che attributi. Ciò è consentito a patto che gli attributi siano presenti nella clausola Group By.

? Per ogni studente, la matricola, il nome ed il numero di esami superati

```
SELECT S.Matricola, S.Nome, count(*)  
as N°Esami  
FROM Studenti S, Esami E  
WHERE S.Matricola = E.Matricola  
GROUP BY S.Matricola, S.Nome
```

Studenti

Matric	Nome	CdL
123	Silvia	SBC
234	Giulia	CMT
345	Silvia	SBC

Vediamo come viene eseguita l'interrogazione:

Consideriamo la giunzione tra Studenti ed Esami

Studenti&Esami

Matric	Nome	CdL	Matric	CodAF	Voto
123	Silvia	SBC	123	AA252	30
123	Silvia	SBC	123	BB112	27
123	Silvia	SBC	123	CC205	22
234	Giulia	CMT	234	BB112	24
345	Silvia	SBC	345	AA252	28
345	Silvia	SBC	345	BB112	27

L'esecuzione della clausola GROUP BY genera i seguenti gruppi

gruppo 123 Silvia

Matric	Nome	CdL	Matric	CodAF	Voto
123	Silvia	SBC	123	AA252	30
123	Silvia	SBC	123	BB112	27
123	Silvia	SBC	123	CC205	22

gruppo 234 Giulia

Matric	Nome	CdL	Matric	CodAF	Voto
234	Giulia	CMT	234	BB112	24

gruppo 345 Silvia

Matric	Nome	CdL	Matric	CodAF	Voto
345	Silvia	SBC	345	AA252	28
345	Silvia	SBC	345	BB112	27

dove sono stati rappresentati con carattere grassetto gli attributi di raggruppamento.

Su ciascun gruppo viene poi calcolata la funzione di aggregazione Count(*) e per ciascun gruppo viene generata un'ennupla del risultato, con gli attributi specificati nella clausola SELECT: **S.Matricola**, **S.Nome**, **N°Esami**, quest'ultimo calcolato come valore di count(*)

Risultato

Matric	Nome	N°Esami
123	Silvia	3
234	Giulia	1
345	Silvia	2

Si osservi che se non avessimo specificato anche Matricola tra gli attributi di raggruppamento, poiché Nome non è chiave avremmo avuto un risultato errato, in quanto il sistema non avrebbe fatto distinzione tra le due studentesse di nome Silvia

```
SELECT S.Nome, count(*), as N°Esami
FROM Studenti S, Esami E
WHERE S.Matricola = E.Matricola
GROUP BY Nome
```

Genera i gruppi

gruppo Silvia

Matric	Nome	CdL	Matric	CodAF	Voto
123	Silvia	SBC	123	AA252	30
123	Silvia	SBC	123	BB112	27
123	Silvia	SBC	123	CC205	22
345	Silvia	SBC	345	AA252	28
345	Silvia	SBC	345	BB112	27

gruppo Giulia

Matric	Nome	CdL	Matric	CodAF	Voto
234	Giulia	CMT	234	BB112	24

Risultato

Nome	N°Esami
Silvia	5
Giulia	1

Attenzione!

Non abusare della GROUP BY, ma utilizzarla solo se necessario.

La clausola **HAVING**

La clausola Having si utilizza solo in connessione con Group BY, e serve a specificare delle condizioni che devono essere soddisfatte dai gruppi.

? **La matricola ed il Numero di crediti acquisiti da ciascuno studente**

```
SELECT E.Matricola, sum(CFU) as N°Crediti
FROM Esami E, AttivitàFormative A
WHERE E.CodiceAF = A.CodiceAF
GROUP BY E.Matricola
```

? La matricola ed il Numero di crediti acquisiti da ciascuno studente di SBC

```
SELECT E.Matricola, sum(CFU) as N°Crediti
FROM  Studenti S, Esami E, AttivitàFormative A
WHERE S.Matricola = E.Matricola
      and E.CodiceAF = A.CodiceAF
      and S.CorsodiLaurea = "SBC"
GROUP BY E.Matricola
```

La condizione S.CorsodiLaurea = "SBC" viene specificata nella clausola WHERE in quanto è espressa su un attributo di Studenti, e riguarda i singoli studenti.

Ma supponiamo di volere:

? La matricola ed il Numero di crediti acquisiti da ciascuno studente che non ha voti inferiori a 27.

La condizione "non ha voti inferiori a 27" è espressa sull'attributo Voto della relazione Esami, ma riguarda l'insieme degli esami sostenuti da uno studente, cioè riguarda i gruppi; non si può pertanto specificare nella clausola WHERE, ma solo dopo il raggruppamento, con la clausola HAVING

```
SELECT E.Matricola, sum(CFU) as N°Crediti
FROM   Esami E, AttivitàFormative A
WHERE  E.CodiceAF = A.CodiceAF
GROUP BY E.Matricola
HAVING min(E.Voto) >= 27
```

La condizione $\min(E.Voto) \geq 27$ viene valutata su ciascun gruppo di esami dello stesso studente.

Consideriamo il solito esempio

AttivitàFormative&Esami

CodAF	NomeAF	CFU	Matric	CodAF	Voto
AA252	Basi di..	5	123	AA252	30
AA252	Basi di..	5	345	AA252	28
BB112	Storia...	5	234	BB112	24
BB112	Storia...	10	123	BB112	27
BB112	Storia...	10	345	BB112	27
CC205	Letter..	10	123	CC205	22

gruppo 123

CodAF	NomeAF	CFU	Matric	CodAF	Voto
AA252	Basi di..	5	123	AA252	30
BB112	Storia...	10	123	BB112	28
CC205	Letter..	10	123	CC205	22

gruppo 234

CodAF	NomeAF	CFU	Matric	CodAF	Voto
BB112	Storia...	5	234	BB112	24

gruppo 345

CodAF	NomeAF	CFU	Matric	CodAF	Voto
AA252	Basi di..	5	345	AA252	28
BB112	Storia...	10	345	BB112	27

Su ciascun gruppo viene valutata la condizione specificata nella clausola HAVING:
 solo il gruppo 345 soddisfa a tale condizione, e quindi è l'unico che rimane per generare il risultato

Risultato

Matric	N°Crediti
345	15

ALTRI PREDICATI

In tutti gli esempi visti sin qui, i predicati presenti nella clausola WHERE sono predicati di confronto:

$$E_1 \text{ cfr } E_2$$

Abbiamo anche visto che il secondo termine del confronto può essere costituito da un comando SELECT, purché abbia come risultato un valore singolo.

Vedremo adesso altri predicati ed altri operatori che estendono le capacità espressive di SQL

Per gli esempi si consideri ancora lo schema

FILM(CodiceVHS, Titolo, Regista, Anno)

ATTORI(Nome, Nazionalità)

RECITA(CodiceVHS*, Nome*, Personaggio)

CASSETTE(Collocazione, CodiceVHS*, DataNoleg,
CodiceCliente*)

CLIENTI(CodiceCliente, Cognome, Nome,
Indirizzo, Telefono)

A IS NULL , A IS NOT NULL

Controlla che l' attributo **A** abbia o non abbia valore nullo

? La collocazione delle cassette non noleggiate

```
SELECT Collocazione  
FROM Cassette  
WHERE CodiceCliente is null
```

? La collocazione delle cassette noleggiate dopo il 1/1/05

```
SELECT Collocazione  
FROM Cassette  
WHERE DataNoleg is not null and  
DataNoleg > 1/1/05
```

L'uso del predicato **is [not] null** è l'unico modo per stabilire se una csassetta è o non è noleggiata.

Nel secondo esempio è necessario controllare che la `DataNoleg` non sia nulla (cioè la cassetta è noleggiata), altrimenti il predicato `DataNoleg > 1/1/05` avrebbe valore indeterminato.

Osserviamo che in effetti in SQL un predicato può assumere, oltre ai valori true e false, anche il valore unknown, in quei casi in cui non può essere valutato (attributi con valore nullo, sottoselect con risultato vuoto).

Per gestire correttamente queste situazioni si utilizza appunto il predicato **is not null**, come nell'esempio precedente

A LIKE maschera

A NOT LIKE maschera

Controlla che il valore dell'attributo **A** sia o non sia conforme alla maschera.

maschera è una sequenza qualunque di caratteri che può contenere i caratteri speciali " - " e " % "

Una parola è conforme alla maschera se

- I caratteri della maschera diversi da – e da % coincidono con quelli della parola.
- Al carattere – nella maschera corrisponde un qualunque carattere della parola
- Al carattere % nella maschera corrisponde una qualunque sequenza, anche vuota, di caratteri nella parola

? CodiceCliente, Cognome e Nome dei Clienti il cui Codice contiene dalla quarta posizione in avanti i caratteri MRC75

```
SELECT CodiceCliente, Cognome, Nome  
FROM Clienti  
WHERE CodF LIKE ---MRC75%
```

ad es. BCEMRC7548, 123MRC75, j23MRC75e6732
sono tutte parole conformi alla maschera ---MRC75%

A BETWEEN E1 AND E2
A NOT BETWEEN E1 AND E2

Controlla che il valore dell'attributo **A** sia o non sia compreso tra **E1** ed **E2**

Non ha grande interesse;

A BETWEEN E1 AND E2

è equivalente a

A >= E1 AND A <= E2

E IN (V1, V2, ...)
E IN (SOTTOSELECT)
E NOT IN (V1, V2, ...)
E NOT IN (SOTTOSELECT)

Controlla che il valore di **E** sia o non sia presente nella lista di valori **V1, V2, ...** oppure nel risultato della SOTTOSELECT

E IN (V1, V2, ...)

è equivalente a

E=V1 or E=V2 or ...

Più interessante l'uso di **NOT IN** con una Sottoselect:

? Il CodiceVHS dei film in cui NON recitano attori francesi.

```
SELECT R.CodiceVHS
FROM Recita R
WHERE "Francia" not in
      (SELECT Nazionalità
       FROM Attori A
       WHERE A.Nome = R.Nome)
```

Osserviamo che l'interrogazione :

? Il CodiceVHS dei film in cui recitano attori francesi.

può essere formulata in modo analogo

```
SELECT R.CodiceVHS
FROM Recita R
WHERE "Francia" in
      (SELECT Nazionalità
       FROM Attori A
       WHERE A.Nome = R.Nome)
```

ma è più semplice, sia per l'utente che per il sistema, la seguente formulazione, che fa uso della giunzione evitando la sottoselect.

```
SELECT R.CodiceVHS
FROM Recita R, Attori A
WHERE A.Nazionalità = "Francia"
      and A.Nome = R.Nome
```

EXISTS (SOTTOSELECT)

Restituisce il valore true se il risultato della sottoselect non è vuoto

NOT EXISTS (SOTTOSELECT)

Restituisce il valore true se il risultato della sottoselect è vuoto

È più utile nella forma negata

? Il CodiceVHS dei film in cui NON recitano attori francesi.

```
SELECT R.CodiceVHS
FROM Recita R
WHERE not exists
      (SELECT *
       FROM Attori A
       WHERE A.Nome = R.Nome
            and A.Nazionalità = "Francia")
```

Osserviamo che, come nel caso precedente, l'interrogazione :
? Il CodiceVHS dei film in cui recitano attori francesi.

può essere formulata in modo analogo

```
SELECT R.CodiceVHS
FROM Recita R
WHERE exists
      (SELECT *
       FROM Attori A
       WHERE A.Nome = R.Nome
            and A.Nazionalità = "Francia")
```

Quantificazione delle interrogazioni

Questi ultimi esempi si prestano ad alcune riflessioni sulla natura delle interrogazioni

Consideriamo le seguenti interrogazioni

- a) I film in cui recitano attori francesi
- b) I film in cui recitano **solo** attori francesi
- c) I film in cui **non** recitano attori francesi

L'interrogazione a) si dice *esistenziale*. Un film appartiene al risultato se esiste **almeno un** attore francese che vi recita, e non importa se ci sono altri attori di altre nazionalità

In SQL si esprime con una SELECT – FROM – WHERE senza necessità di sottoselect

Le interrogazioni b) e c) si dicono *universali*. Un film appartiene al risultato se **tutti** i gli attori che vi recitano sono francesi (interr b)) o se **nessuno** è francese (interr. c)).

Per esprimerle in SQL è necessario far ricorso alla sottoselect

Consideriamo un altro esempio

d) Matricola e Nome degli studenti che hanno voti inferiori a 27

Si tratta di una interrogazione esistenziale, basta che esista un voto dello studente inferiore a 27

```
SELECT DISTINCT S.Matricola, S.Nome
FROM   Studenti S, Esami E
WHERE  S.Matricola = E.Matricola
       and Voto < 27
```

Consideriamo la solita istanza della BD

Studenti

Matric	Nome	CdL
123	Silvia	SBC
234	Giulia	CMT
345	Silvia	SBC

Esami

Matric	CodAF	Voto
123	AA252	30
234	BB112	24
345	AA252	28
123	BB112	27
123	CC205	22
345	BB112	27

Risultato d)

Matric	Nome	CdL
123	Silvia	SBC
234	Giulia	CMT

e) **Matricola e Nome degli studenti che non hanno voti inferiori a 27**

Si tratta di una interrogazione universale

e1)

```
SELECT S.Matricola, S.Nome
FROM  Studenti S
WHERE not exists
      (SELECT *
       FROM Esami E
       WHERE S.Matricola = E.Matricola
            and Voto < 27)
```

Risultato e1)

Matric	Nome
345	Silvia

Se avessimo scritto, per voler fare a meno della sottoselect

e2)

```
SELECT DISTINCT S.Matricola, S.Nome
FROM  Studenti S, Esami E
WHERE S.Matricola = E.Matricola
      and not(Voto < 27)
```

avremmo avuto

Risultato e2)

Matric	Nome
345	Silvia
123	Silvia

in quanto 123 Silvia ha voti superiori a 27, ma poiché ha anche voti inferiori a 27 non soddisfa alla condizione espressa nella e).

Osserviamo che la interrogazione e1) restituisce anche gli studenti che non hanno dato alcun esame.

f) Matricola e Nome degli studenti che hanno tutti i voti inferiori a 27

Si tratta anche questa di una interrogazione universale

```
SELECT S.Matricola, S.Nome
FROM Studenti
WHERE not exists
      (SELECT *
       FROM Esami E
       WHERE S.Matricola = E.Matricola
            and Voto >= 27)
```

è identica alla e), basta negare la condizione. (voto \geq 27 è equivalente a $\text{not}(\text{voto} < 27)$)-

In altre parole, prima di scrivere in SQL la

Matricola e Nome degli studenti che
hanno tutti i voti inferiori a 27

l'abbiamo trasformata nella forma equivalente

Matricola e Nome degli studenti che
non hanno voti maggiori o eguali a 27

In altre parole ancora, abbiamo trasformato la condizione

Ogni voto è inferiore a 27

nella forma equivalente

Non esistono voti maggiori o eguali a 27.

Questa tecnica consente di esprimere qualunque interrogazione universale con la struttura

```
SELECT Attributi
FROM Relazioni1
WHERE not exists
      ( SELECT *
        FROM Relazioni2
        WHERE ....)
```

Relazioni1 hanno gli attributi del risultato, Relazioni2 hanno gli attributi della condizione. Nella clausola Where interna deve esser presente anche la condizione di giunzione fra Relazioni2 e Relazioni1.

Chiariremo questo discorso con altri esempi.

Atri Predicati

SQL offre altri predicati, in realtà ridondanti perché non aggiungono capacità di calcolo ma solo espressività.

E cfr ALL (SOTTOSELECT)
E cfr ANY (SOTTOSELECT)

ove come al solito *E* è un'espressione e *cfr* rappresenta un operatore di confronto.

In questo modo si ha la possibilità di confrontare il valore di un'espressione con un insieme di valori ottenuti come risultato di una sottoselect.

E cfr ALL (SOTTOSELECT)

ha valore **true** se l'espressione sta nella relazione specificata con tutti gli elementi appartenenti al risultato della sottoselect

E cfr ANY (SOTTOSELECT)

ha valore **true** se l'espressione sta nella relazione specificata con almeno uno degli elementi appartenenti al risultato della sottoselect

Possono essere utili per esprimere interrogazioni universali.

```
SELECT S.Matricola, S.Nome  
FROM Studenti  
WHERE 27 >= all  
      (SELECT E.Voto  
       FROM Esami E  
       WHERE S.Matricola = E.Matricola )
```

Restituisce Matricola e Nome degli studenti che hanno tutti i voti inferiori a 27; è un altro modo di formulare la f)

```
SELECT S.Matricola, S.Nome
FROM Studenti
WHERE 27 >= any
      (SELECT E.Voto
       FROM Esami E
       WHERE S.Matricola = E.Matricola )
```

Restituisce Matricola e Nome degli studenti che hanno almeno un voto inferiore a 27; è un altro modo di formulare la d), ma si raccomanda di non utilizzarlo

Per concludere, ribadiamo che è necessario distinguere tra

- Interrogazione **esistenziale**: si esprime in SQL con una interrogazione di giunzione (senza sottoselect) e
- interrogazione **universale**: si esprime in SQL con predicati **not in** , **not exists**, cfr **all** seguiti da un **sottoselect**.

Osserviamo che, nel caso di interrogazioni universali talvolta risulta più semplice formularla in forma negata, e poi esprimerla in SQL con il predicato **not exists**

ESEMPI

Riprendiamo lo schema

FILM(CodiceVHS, Titolo, Regista, Anno)

ATTORI(Nome, Nazionalità)

RECITA(CodiceVHS*, Nome*, Personaggio)

consideriamo le seguenti interrogazioni

- A. Gli attori italiani che hanno recitato nei film di Fellini
- B. Gli attori italiani che hanno recitato solo nei film di Fellini
- C. Per ogni attore, l'anno di produzione del primo film di Fellini in cui ha recitato

A. Gli attori italiani che hanno recitato nei film di Fellini

Si tratta di una interrogazione esistenziale:

```
SELECT A.Nome  
FROM Attori A, Recita R, Film F  
WHERE A.Nome = R.Nome  
      and R.CodiceVHS = F.CodiceVHS  
      and F.Regista = "Fellini"
```

B. Gli attori italiani che hanno recitato solo nei film di Fellini

Si tratta di una interrogazione universale:

```
SELECT A.Nome  
FROM Attori A  
WHERE "Fellini" = all  
      (SELECT F.Regista  
       FROM Film F, Recita R  
       WHERE A.Nome = R.Nome  
             and R.CodiceVHS = F.CodiceVHS)
```

B' Gli attori italiani che **non** hanno recito in film con regista **diverso** da Fellini

```
SELECT A.Nome
FROM Attori A
WHERE not exists
  (SELECT *
   FROM Film F, Recita R
   WHERE A.Nome = R.Nome
        and R.CodiceVHS = F.CodiceVHS
        and F.Regista < > "Fellini")
```

C. Per ogni attore, l'anno di produzione del primo film di Fellini in cui ha recitato

Questa è un'interrogazione esistenziale, che però richiede un raggruppamento

```
SELECT A.Nome, min(F.Anno) as AnnoPrimoFilm
FROM Attori A, Recita R, Film F
WHERE A.Nome = R.Nome
      and R.CodiceVHS = F.CodiceVHS
      and F.Regista = "Fellini"
GROUP BY A.Nome
```

Se qualcuno si sente particolarmente bravo, scriva in SQL la seguente interrogazione

A. Gli attori italiani che hanno recitato in tutti i film di Fellini