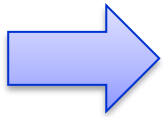
LA QUANTIFICAZIONE UNIVERSALE

- Gli studenti che hanno preso solo 30
- Errore comune (e grave):

```
SELECT s.Nome
FROM Studenti s, Esami e
WHERE e.Matricola = s.Matricola AND e.Voto = 30
```



LA QUANTIFICAZIONE UNIVERSALE CON ALL

- la query (studenti con tutti 30):

```
SELECT s.Nome FROM Studenti s
WHERE NOT EXISTS (SELECT * FROM Esami e
                  WHERE e.Matricola = s.Matricola
                  AND e.Voto <> 30)
```

Funziona?

- Sostituendo EXISTS con =ANY, diventa:

```
SELECT s.Nome FROM Studenti s
WHERE NOT (s.Matricola =ANY (SELECT e.Matricola FROM Esami e
                             WHERE e.Voto <> 30))
```

- Ovvero:

```
SELECT s.Nome FROM Studenti s
WHERE s.Matricola <>ALL (SELECT e.Matricola FROM Esami e
                         WHERE e.Voto <> 30)
```

- Naturalmente, <>ALL è lo stesso di NOT IN...

LA QUANTIFICAZIONE UNIVERSALE E GLI INSIEMI VUOTI

- Trovare gli studenti che hanno preso solo trenta:

```
SELECT s.Nome
FROM Studenti s
WHERE NOT EXISTS (SELECT *
                  FROM Esami e
                  WHERE e.Matricola = s.Matricola AND e.Voto <> 30)
```

- Perché trovo anche Rossi?

Nome	Matricola	Provincia	AnnoNascita
Bianco	1	PI	1996
Verdi	2	PI	1992
Rossi	3	PI	1992

Mater.	Matricola	Voto
RC	1	30
IS	2	30
RC	2	20

LA QUANTIFICAZIONE UNIVERSALE E GLI INSIEMI VUOTI

Studenti

MATRICOLA	NOME	COGNOME	DATAISCRIZIONE
282320	Gianna	Neri	04-MAG-20
369871	Mario	Rossi	04-MAG-20
515140	Mario	Verdi	(null)
090456	Mario	Bianchi	04-MAG-20
579555	Luigi	Rossi	(null)
018701	Luca	Bianchi	04-MAG-20
100100	Sara	Viola	04-GIU-20

Esami

CODICE	MATRIC	VOTO	DATAESAME
A	369871	30	04-MAG-20
B	369871	29	10-MAG-21
C	369871	30	10-GIU-21
D	369871	27	20-GIU-21
A	515140	30	15-MAG-21
B	515140	28	12-MAG-21
C	090456	27	13-MAG-21
D	090456	26	14-GIU-21
A	018701	30	22-LUG-21
D	282320	20	12-GEN-21

Trovare gli studenti che hanno preso solo trenta:

```
SELECT s.Nome, s.cognome, s.matricola
FROM Studenti s
WHERE NOT EXISTS (SELECT * FROM Esami e
        WHERE e. Matricola = s.Matricola
                AND e.Voto <> 30)
```

NOME	COGNOME	MATRICOLA
Luigi	Rossi	579555
Sara	Viola	100100
Luca	Bianchi	018701

Cosa cambia se invece di **NOT EXISTS** uso **<>ALL** oppure **NOT IN**?

LA QUANTIFICAZIONE UNIVERSALE E GLI INSIEMI VUOTI

Studenti

MATRICOLA	NOME	COGNOME	DATAISCRIZIONE
282320	Gianna	Neri	04-MAG-20
369871	Mario	Rossi	04-MAG-20
515140	Mario	Verdi	(null)
090456	Mario	Bianchi	04-MAG-20
579555	Luigi	Rossi	(null)
018701	Luca	Bianchi	04-MAG-20
100100	Sara	Viola	04-GIU-20

Esami

CODICE	MATRIC	VOTO	DATAESAME
A	369871	30	04-MAG-20
B	369871	29	10-MAG-21
C	369871	30	10-GIU-21
D	369871	27	20-GIU-21
A	515140	30	15-MAG-21
B	515140	28	12-MAG-21
C	090456	27	13-MAG-21
D	090456	26	14-GIU-21
A	018701	30	22-LUG-21
D	282320	20	12-GEN-21

Trovare gli studenti che hanno preso solo trenta:

```
SELECT s.Nome, s.cognome, s.matricola
FROM Studenti s
WHERE s.Matricola <> ALL (SELECT e.Matricola
                          FROM Esami e
                          WHERE e.Voto <> 30)
```

NOME	COGNOME	MATRICOLA
Luigi	Rossi	579555
Sara	Viola	100100
Luca	Bianchi	018701

LA QUANTIFICAZIONE UNIVERSALE E GLI INSIEMI VUOTI

Studenti

MATRICOLA	NOME	COGNOME	DATAISCRIZIONE
282320	Gianna	Neri	04-MAG-20
369871	Mario	Rossi	04-MAG-20
515140	Mario	Verdi	(null)
090456	Mario	Bianchi	04-MAG-20
579555	Luigi	Rossi	(null)
018701	Luca	Bianchi	04-MAG-20
100100	Sara	Viola	04-GIU-20

Esami

CODICE	MATRIC	VOTO	DATAESAME
A	369871	30	04-MAG-20
B	369871	29	10-MAG-21
C	369871	30	10-GIU-21
D	369871	27	20-GIU-21
A	515140	30	15-MAG-21
B	515140	28	12-MAG-21
C	090456	27	13-MAG-21
D	090456	26	14-GIU-21
A	018701	30	22-LUG-21
D	282320	20	12-GEN-21

Trovare gli studenti che hanno preso solo trenta:

```
SELECT s.Nome, s.cognome, s.matricola
FROM Studenti s
WHERE s.Matricola NOT IN (SELECT e.Matricola
                          FROM Esami e
                          WHERE e.Voto <> 30)
```

NOME	COGNOME	MATRICOLA
Luigi	Rossi	579555
Sara	Viola	100100
Luca	Bianchi	018701

LA QUANTIFICAZIONE UNIVERSALE E GLI INSIEMI VUOTI

Studenti

MATRICOLA	NOME	COGNOME	DATAISCRIZIONE
282320	Gianna	Neri	04-MAG-20
369871	Mario	Rossi	04-MAG-20
515140	Mario	Verdi	(null)
090456	Mario	Bianchi	04-MAG-20
579555	Luigi	Rossi	(null)
018701	Luca	Bianchi	04-MAG-20
100100	Sara	Viola	04-GIU-20

Esami

CODICE	MATRIC	VOTO	DATAESAME
A	369871	30	04-MAG-20
B	369871	29	10-MAG-21
C	369871	30	10-GIU-21
D	369871	27	20-GIU-21
A	515140	30	15-MAG-21
B	515140	28	12-MAG-21
C	090456	27	13-MAG-21
D	090456	26	14-GIU-21
A	018701	30	22-LUG-21
D	282320	20	12-GEN-21

Trovare gli studenti che hanno preso solo trenta:

```
SELECT s.Nome, s.cognome, s.matricola
FROM Studenti s
WHERE NOT EXISTS (SELECT * FROM Esami e
        WHERE e. Matricola = s.Matricola
                AND e.Voto <> 30)
AND EXISTS (SELECT * FROM Esami e
        WHERE e. Matricola = s.Matricola)
```

NOME	COGNOME	MATRICOLA
Luca	Bianchi	018701

GLI INSIEMI VUOTI

- Se voglio gli studenti che hanno preso solo più di 27, e hanno superato qualche esame:

```
SELECT s.Nome
FROM Studenti s
WHERE NOT EXISTS (SELECT *
                  FROM Esami e
                  WHERE e. Matricola = s.Matricola AND e.Voto < 27)
      AND EXISTS (SELECT *
                  FROM Esami e
                  WHERE e. Matricola = s.Matricola)
```

- Oppure:

```
SELECT s.Nome
 FROM Studenti s, Esami e
 WHERE s.Matricola = e.Matricola
 GROUP BY s.Matricola, s. Nome
 HAVING Min(e.Voto) >= 27
```

OTTIMIZZARE IL NOT EXISTS

- Loop interno valutato una volta per ogni studente:

```
SELECT s.Nome
FROM Studenti s
WHERE NOT EXISTS (SELECT *
                  FROM Esami e
                  WHERE e.Matricola = s.Matricola AND e.Voto >
21)
```

- Loop interno valutato una sola volta (loop interno *decorrelato*):

```
SELECT s.Nome
FROM Studenti s
WHERE s.Matricola NOT IN (SELECT e.Matricola
                          FROM Esami e
                          WHERE e.Voto > 21)
```

NOT IN ED OUTER JOIN

- Loop interno valutato una sola volta (loop interno *decorrelato*):

```
SELECT s.Nome
FROM Studenti s
WHERE s.Matricola NOT IN (SELECT e.Matricola
                          FROM Esami e
                          WHERE e.Voto > 21)
```

- Outer join:

```
SELECT s.Nome
FROM Studenti s LEFT JOIN (SELECT *
                           FROM Esami e
                           WHERE e.Voto > 21) USING Matricola
WHERE e.Voto IS NULL
```

Subquery nel calcolo di espressioni

Le subquery oltre che essere usate all'interno della clausola WHERE, possono anche essere utilizzate nel calcolo di espressioni, dunque per definire colonne.

Esempio: Per ogni veicolo rappresentare la targa, la cilindrata e la differenza fra la cilindrata e la cilindrata minima

Veicoli

Targa	Cod_mod	Categoria	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm

```
Select Targa, Cilindrata, Cilindrata – (Select min (Cilindrata)
                                        From Veicoli) Differenza
From Veicoli
```

Esempio

Veicoli

Targa	Cod_mod*	Categoria	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm

Modelli

Cod_Mod	Nome_Mod	Cod_Fab	Cilind_Media

Per ciascun veicolo rappresentare la targa e la differenza fra la cilindrata e la cilindrata media del proprio modello

```
Select targa, Cilindrata – (Select cilind_media
                            From Modelli
                            Where Veicoli.cod_mod=Modelli.cod_mod)
From Veicoli
```

Tabella select principale

UNIONE, INTERSEZIONE, DIFFERENZA

Operazioni booleane su tabelle

A volte può essere utile poter ottenere un'unica tabella contenente alcuni dei dati contenuti in due tabelle omogenee, ossia con attributi definiti sullo stesso dominio. Per esempio date le tabelle:

Paternità

PADRE	FIGLIO
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Maternità

MADRE	FIGLIO
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Operazioni booleane, calcolando unione, intersezione e differenza.

Unione, intersezione e differenza

In SQL la SELECT da sola non permette di fare questo tipo di operazioni su tabelle. Esistono per questo dei costrutti espliciti che utilizzano le parole chiave

UNION
INTERSECT
EXCEPT (in oracle MINUS)

Tali operatori lavorano sulle tabelle come se fossero insiemi di righe, dunque i duplicati vengono eliminati (a meno che si usi la specifica ALL) anche dalle proiezioni!

Unione

L'operatore UNION realizza l'operazione di unione definita nell'algebra relazionale. Utilizza come operandi le due tabelle risultanti da comandi SELECT e restituisce una terza tabella che contiene tutte le righe della prima e della seconda tabella. Nel caso in cui dall'unione e dalla proiezione risultassero delle righe duplicate, l'operatore UNION ne mantiene una sola copia, a meno che non sia specificata l'opzione ALL che indica la volontà di mantenere i duplicati

```
SELECT ...
UNION [all]
SELECT ...
```

Notazione posizionale!

```
SELECT padre
FROM paternita
UNION
SELECT madre
FROM maternita
```

- quali nomi per gli attributi del risultato?
  - nessuno
  - quelli del primo operando
  - ...

In SQL quelli del primo operando

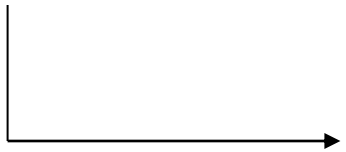
Paternità

PADRE	FIGLIO
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Maternità

MADRE	FIGLIO
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

```
SELECT padre, figlio
FROM paternita
UNION
SELECT madre, figlio
FROM maternita
```



L'attributo Padre della prima Select viene messo nella stessa colonna con l'attributo Madre della seconda select

PADRE	FIGLIO
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Paternità

PADRE	FIGLIO
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Maternità

MADRE	FIGLIO
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

```
SELECT Padre, Figlio
FROM paternita
UNION
SELECT Figlio, Madre
FROM maternita

SELECT padre as genitore, figlio
FROM paternita
UNION
SELECT figlio, madre as genitore
FROM maternita
```

L'attributo Padre della prima Select viene messo nella stessa colonna con l'attributo Figlio della seconda select

PADRE	FIGLIO
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo
Maria	Luisa
Luigi	Luisa
Olga	Anna
Filippo	Anna
Andrea	Maria
Aldo	Maria

Riepilogo: Notazione posizionale

Sbagliata

```
SELECT padre as genitore, figlio
FROM paternita
UNION
SELECT figlio, madre as genitore
FROM maternita
```

Corretta:

```
SELECT padre as genitore, figlio
FROM paternita
UNION
SELECT madre as genitore, figlio
FROM maternita
```

Quanto detto riguardo alla notazione posizionale dell'operatore UNION vale equivalentemente per gli altri operatori booleani EXCEPT (Minus in Oracle) e INTERSECT.

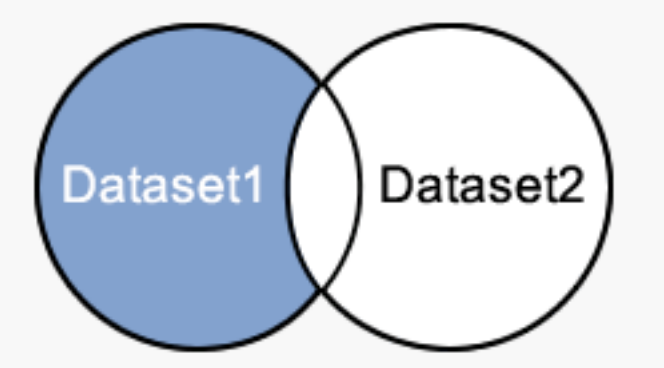
Differenza

L'operatore EXCEPT utilizza come operandi due tabelle ottenute mediante due select e ha come risultato una nuova tabella che contiene tutte le righe della prima che non si trovano nella seconda.

Realizza la differenza dell'algebra relazionale. Anche qui si può specificare l'opzione ALL per indicare la volontà di mantenere i duplicati.

Sintassi:

```
SELECT ...
EXCEPT [ALL]
SELECT ...
```



Except, esempio

Impiegati il cui nome che non coincide col cognome di qualche altro impiegato

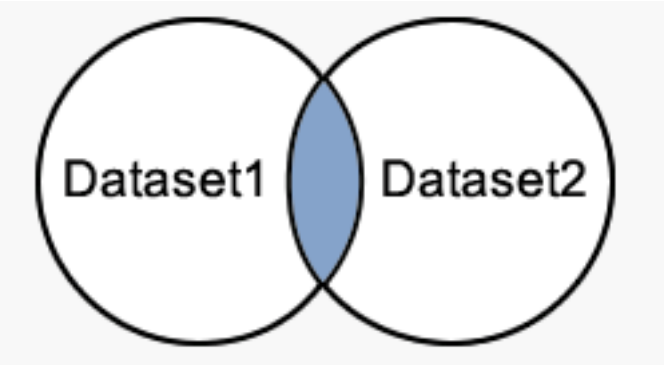
```
SELECT Nome
FROM Impiegato
EXCEPT
SELECT Cognome as Nome
FROM Impiegato
```

Nota che la differenza può essere effettuata da un'unica select che utilizza subselect

```
SELECT nome
FROM Impiegato
WHERE nome<>All[not in]
                (SELECTcognome
                 FROM Impiegato)
```

Intersezione

L'operatore INTERSECT utilizza come operandi due tabelle risultanti dai comandi SELECT e restituisce una tabella che contiene le righe comuni alle due tabelle iniziali. Realizza l'intersezione dell'algebra relazionale.



Sintassi
SELECT …
INTERSECT [ALL]
SELECT …

L'opzione ALL serve a mantenere gli eventuali duplicati di righe.
In sua assenza, si mantiene una sola copia delle righe duplicate.

SQL PER LA MODIFICA DI BASI DI DATI

Data Manipulation Language

Introduciamo ora il Data Manipulation Language (DML) ossia il linguaggio SQL che serve per inserire, modificare e cancellare i dati del database, ma anche per interrogare il database, ossia estrarre i dati dal database.

Inizialmente descriveremo le istruzioni che servono a inserire, cancellare e modificare i dati.

In seguito introdurremo le istruzioni per estrarre dal database le informazioni che ci interessano.

SQL PER MODIFICARE I DATI

- INSERT INTO Tabella [(A1,..,An)]
  (VALUES (V1,..,Vn) | AS Select)

- UPDATE Tabella
  SET Attributo = Expr, …, Attributo = Expr
  WHERE Condizione

- DELETE FROM Tabella
  WHERE Condizione

Insert semplice

- Supponiamo di volere inserire un nuovo dato in una tabella.
- Tale operazione si realizza mediante l'istruzione

INSERT INTO… VALUES

Sintassi:

INSERT INTO nome tabella
[(ListaAttributi)] VALUES (ListaDiValori) |
SQLSelect

Insert, Esempio 1

Supponiamo di avere definito la tabella:

Esami

Corso	Insegnante	Matricola	Voto

```
INSERT INTO Esami (Corso, Matricola, Voto)
VALUES ('DB1', '123456', 27)
```

Corso	Insegnante	Matricola	Voto
DB1	NULL	123456	27

Ai valori non attribuiti viene assegnato NULL, a meno che non sia specificato un diverso valore di default.
L'inserimento fallisce se NULL non è permesso per gli attributi mancanti.

Insert, Esempio 2

```
INSERT INTO Esami
VALUES ('DB2', 'Verdi', '123123', 30)
```

Corso	Insegnante	Matricola	Voto
DB1	NULL	123456	27
DB2	Verdi	123123	30

Non specificare gli attributi equivale a specificare tutte le colonne della tabella.
NB: Si deve rispettare l'ordine degli attributi

Insert mediante SELECT

E' possibile effettuare un insert prendendo i dati da un'altra tabella. Questo è possibile mediante il comando di interrogazione del database SELECT, che vedremo nelle prossime lezioni. Parleremo in seguito di questo tipo di inserimento di dati.
Con questo tipo di insert si possono effettuare tanti inserimenti simultaneamente.
Esempio:
Tabella: Indirizzi_Studenti(Indirizzo, Telefono, Email)

```
INSERT INTO Indirizzi_Studenti (Indirizzo, Telefono)
SELECT Indirizzo, Telefono
FROM Studenti
```

Delete

Cancellazione di righe da tabelle

Sintassi:

```
DELETE FROM nome_tabella
[WHERE Condizione]
```

Per eliminare un elemento bisogna individuare quale.
Questo si può stabilire mediante la clausola WHERE, dove viene stabilita una condizione che individua l'elemento (o gli elementi) da cancellare.
Spesso un particolare elemento può essere individuato mediante il suo valore nella chiave primaria.

Delete, esempio

Cancellare dalla tabella Esami i dati relativi allo studente il cui numero di matricola è '123456'

```
DELETE FROM Esami
WHERE Matricola = '123456'
```

Operazione non reversibile
Se la condizione è omessa questa istruzione cancella l'intero contenuto della tabella
(!!! Attenzione quindi !!!)

Lo schema invece non viene modificato. L'istruzione

```
DELETE FROM Esami
```

Restituisce la tabella Esami con l'istanza vuota

Delete

- La condizione del delete può essere una normale condizione di SELECT (vedremo dopo)
- Questa modalità di delete permette di cancellare più righe con un'unica istruzione, purchè le righe soddisfinola condizione.
- Esempio:
  - Eliminare tutte le righe della tabella esami in cui il numero di matricola non si trova nella tabella Studenti

```
DELETE FROM Esami
WHERE Matricola NOT IN
(SELECT Matricola FROM Studenti)
```

Update

Inoltre è possibile aggiornare alcuni dati seguendo la seguente sintassi:

```
UPDATE Tabella
SET Attributo = Espr
WHERE Condizione
```

Update, esempi

Esempio: Dalla tabella Aule modificare il numero dell'aula da 3 a 7.

```
UPDATE Aule
SET Aula = 7
WHERE Aula = 3
```

Esempio: modificare il valore del reddito delle persone più giovani di 30 anni attribuendo loro un aumento del 10%.

```
UPDATE Persone
SET Reddito = Reddito * 1.1
WHERE Eta < 30
```