Il linguaggio SQL

- Interrogazioni -

Dott. Nicola Dragoni
nicola.dragoni@gmail.com

(Il contenuto di queste slide è stato originariamente creato dal Dott. Matteo Magnani)

Il linguaggio SQL

- Il linguaggio SQL (Structured Query Language) è il linguaggio standard per la definizione, manipolazione e interrogazione delle basi di dati relazionali

- Il linguaggio SQL è stato originariamente sviluppato nei laboratori di ricerca IBM per il sistema relazionale System R

DB-Persone

Persone

Nome	*Eta*	*Reddito*
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Maternita

Madre	*Figlio*
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Paternita

Padre	*Figlio*
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Interrogazione semplice

Nome e Reddito delle persone con meno di 30 anni

Elementi dell'interrogazione

Nome e Reddito delle persone con meno di 30 anni.

Tabella/e da utilizzare

Nome	*Eta*	*Reddito*
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Persone

Elementi dell'interrogazione

Nome e Reddito delle persone con meno di 30 anni.

Condizione

Tabella/e da utilizzare

Nome	*Eta*	*Reddito*
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Persone

Elementi dell'interrogazione

Attributi su cui proiettare il risultato

Nome e **Reddito** delle **persone** con **meno di 30 anni**.

Condizione

Tabella/e da utilizzare

Nome	*Eta*	*Reddito*
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Persone

Interrogazione semplice in SQL

```
SELECT Nome, Reddito
FROM Persone
WHERE Eta < 30
```

→ Target list

→ Clausola From

→ Clausola Where

Semantica di una query SELECT/FROM/WHERE

1. Si esegue il prodotto cartesiano delle tabelle coinvolte (in questo caso, essendoci una sola tabella, il prodotto cartesiano non viene effettuato)

2. Si selezionano le righe (tuple) sulla base del predicato della clausola Where

3. Si proietta sugli attributi della target list

```
SELECT Nome, Reddito   ⟶ Target list
FROM Persone           ⟶ Clausola From
WHERE Eta < 30         ⟶ Clausola Where
```

(1) Prodotto cartesiano (1 tabella)

```
SELECT Nome, Reddito
FROM Persone
WHERE Eta < 30
```

Nome	*Eta*	*Reddito*
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Persone

(1b) Prodotto cartesiano (2 tabelle)

Padre	Figlio
Sergio	Franco
Luigi	Olga

Pat

⋈

Madre	Figlio
Luisa	Maria
Luisa	Luigi
Anna	Olga

Mat

=

Padre	Figlio	Madre	Figlio1
Sergio	Franco	Luisa	Maria
Sergio	Franco	Luisa	Luigi
Sergio	Franco	Anna	Olga
Luigi	Olga	Luisa	Maria
Luigi	Olga	Luisa	Luigi
Luigi	Olga	Anna	Olga

(2) Selezione

```
SELECT Nome, Reddito
FROM Persone
WHERE Eta < 30
```

Nome	*Eta*	*Reddito*
Andrea	**27**	**21**
Aldo	**25**	**15**
Maria	55	42
Anna	50	35
Filippo	**26**	**30**
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Persone

(3) Proiezione

```
SELECT Nome, Reddito
FROM Persone
WHERE Eta < 30
```

Nome	*Eta*	*Reddito*
Andrea	27	**21**
Aldo	25	**15**
Maria	55	42
Anna	50	35
Filippo	26	**30**
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Persone

Risultato

```
SELECT Nome, Reddito
FROM Persone
WHERE Eta < 30
```

Nome	*Reddito*
Andrea	21
Aldo	15
Filippo	30

DB-Impiegati

Nome	*Cognome*	*Dipart*	*Ufficio*	*Stipendio*
Mario	Rossi	Amministrazione	10	45
Carlo	Bianchi	Produzione	20	36
Giuseppe	Verdi	Amministrazione	20	40
Franco	Neri	Distribuzione	16	45
Carlo	Rossi	Direzione	14	80
Lorenzo	Lanzi	Direzione	7	73
Paola	Borroni	Amministrazione	75	40
Marco	Franco	Produzione	20	46

Impiegato

Nome	*Indirizzo*	*Citta*
Amministrazione	Via Tito Livio	Milano
Produzione	Piazza Lavater	Torino
Distribuzione	Via Segre	Roma
Direzione	Via Tito Livio	Milano
Ricerca	Via Morone	Milano

Dipartimento

Target list: selezione senza proiezione

```
SELECT *
FROM Impiegato
WHERE Cognome = 'Rossi'
```

Nome	*Cognome*	*Dipart*	*Ufficio*	*Stipendio*
Mario	Rossi	Amministrazione	10	45
Carlo	Rossi	Direzione	14	80

Target list: selezione con proiezione

```
SELECT Nome, Cognome, Stipendio
FROM Impiegato
WHERE Cognome = 'Rossi'
```

Nome	*Cognome*	*Stipendio*
Mario	Rossi	45
Carlo	Rossi	80

Target list: proiezione senza selezione

```
SELECT Nome, Cognome
FROM Impiegato
```

Nome	*Cognome*
Mario	Rossi
Carlo	Bianchi
Giuseppe	Verdi
Franco	Neri
Carlo	Rossi
Lorenzo	Lanzi
Paola	Borroni
Marco	Franco

Target list: proiezione con/senza duplicati

```
SELECT Cognome
FROM Impiegato
```

Cognome
Rossi
Bianchi
Verdi
Neri
Rossi
Lanzi
Borroni
Franco

```
SELECT DISTINCT Cognome
FROM Impiegato
```

Cognome
Rossi
Bianchi
Verdi
Neri
Lanzi
Borroni
Franco

Target list: espressioni

```
SELECT Stipendio/12 As StipendioMensile
FROM Impiegato
WHERE Cognome = ‘Bianchi’
```

StipendioMensile
3.00

Clausola WHERE: disgiunzione

```
SELECT Nome, Cognome
FROM Impiegato
WHERE Dipart = 'Amministrazione' OR
      Dipart = 'Produzione'
```

Nome	*Cognome*
Mario	Rossi
Carlo	Bianchi
Giuseppe	Verdi
Paola	Borroni
Marco	Franco

Clausola WHERE: condizione complessa

```
SELECT Nome
FROM Impiegato
WHERE Cognome = ‘Rossi’ AND
        (Dipart = ‘Amministrazione’ OR
Dipart = ‘Produzione’)
```

Nome
Mario

Clausola WHERE: operatore IN

```
SELECT Nome
FROM Impiegato
WHERE Cognome = ‘Rossi’ AND
        Dipart IN (‘Amministrazione’,
                   ‘Produzione’)
```

Nome
Mario

Clausola WHERE: operatore LIKE

```
SELECT *
FROM Impiegato
WHERE Cognome LIKE '_o%i'
```

_ → Un carattere qualsiasi

% → Un stringa qualsiasi

Nome	*Cognome*	*Dipart*	*Ufficio*	*Stipendio*
Mario	Rossi	Amministrazione	10	45
Carlo	Rossi	Direzione	14	80
Paola	Borroni	Amministrazione	75	40

Clausola WHERE: operatore BETWEEN

```
SELECT *
FROM Impiegato
WHERE Stipendio BETWEEN 40 AND 45
```

Nome	*Cognome*	*Dipart*	*Ufficio*	*Stipendio*
Mario	Rossi	Amministrazione	10	45
Giuseppe	Verdi	Amministrazione	20	40
Franco	Neri	Distribuzione	16	45
Paola	Borroni	Amministrazione	75	40

Clausola WHERE: valori nulli

"Impiegati che hanno o potrebbero avere uno stipendio minore di 50"

- N.B.: Vogliamo anche gli stipendi "nulli"

Nome	*Cognome*	*Dipart*	*Ufficio*	*Stipendio*
Mario	Rossi	Amministrazione	10	45
Carlo	Rossi	Direzione	14	80
Paola	Borroni	Amministrazione	75	***null***

Impiegati_con_nulli

Clausola WHERE: valori nulli

```
SELECT *
FROM Impiegati_con_nulli
WHERE Stipendio < 50 or Stipendio IS NULL
```

Nome	***Cognome***	***Dipart***	***Ufficio***	***Stipendio***
Mario	Rossi	Amministrazione	10	45
Paola	Borroni	Amministrazione	75	***null***

Ordinamento del risultato

- A differenza del modello relazionale, in cui le tuple non sono ordinate, le righe di una tabella possono esserlo - anche se solo al momento della presentazione all'utente.

- Talvolta la possibilità di ordinare il risultato di un'interrogazione è importante. Ad esempio, se si voglio gli stipendi in ordine dal minore al maggiore.

- SQL mette a disposizione la clausola ORDER BY

Ordinamento del risultato: esempio

```
SELECT Cognome, Nome, Stipendio
FROM Impiegato
WHERE Dipartimento LIKE 'Amm%'
ORDER BY Stipendio DESC, Cognome ASC
```

discendente

ascendente (default)

Cognome	***Nome***	***Stipendio***
Rossi	Mario	45
Borroni	Paola	40
Verdi	Giuseppe	40

JOIN Implicito

- Il JOIN è un operatore fondamentale, in quanto permette di utilizzare congiuntamente le informazioni contenute in più tabelle

- Un JOIN corrisponde a un prodotto cartesiano seguito da una selezione

- E’ quindi possibile realizzare un JOIN tramite gli statement SQL visti finora, cioè FROM e WHERE, che permettono di compiere prodotti cartesiani e selezioni

- Esistono anche operatori specifici, ma non li vedremo

DB-Persone

Persone

Nome	*Eta*	*Reddito*
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Maternita

Madre	*Figlio*
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Paternita

Padre	*Figlio*
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Prodotto cartesiano

- Il prodotto cartesiano di due o più tabelle si ottiene riportando le tabelle nella clausola From, senza clausola Where

```
SELECT *
FROM Pat, Mat
```

Padre	Figlio
Sergio	Franco
Luigi	Olga

Pat

⋈

Madre	Figlio
Luisa	Maria
Luisa	Luigi
Anna	Olga

Mat

=

Padre	Figlio	Madre	Figlio1
Sergio	Franco	Luisa	Maria
Sergio	Franco	Luisa	Luigi
Sergio	Franco	Anna	Olga
Luigi	Olga	Luisa	Maria
Luigi	Olga	Luisa	Luigi
Luigi	Olga	Anna	Olga

JOIN Implicito

- Query: “Padre e madre di ogni persona”

Padre	*Figlio*
Sergio	Franco
Luigi	**Olga**
Luigi	Filippo
Franco	Andrea
Franco	Aldo

⋈

Madre	*Figlio*
Luisa	Maria
Luisa	Luigi
Anna	**Olga**
Anna	Filippo
Maria	Andrea
Maria	Aldo

=

Padre	*Figlio*	*Madre*
Luigi	***Olga***	**Anna**
Luigi	Filippo	Anna
Franco	Andrea	Maria
Franco	Aldo	Maria

```
SELECT Padre, Paternita.Figlio, Madre
FROM Paternita, Maternita
WHERE Paternita.Figlio = Maternita.Figlio
```

Esempio: Selezione, Proiezione e JOIN

- Query: “I padri di persone che guadagnano più di venti milioni”

```
SELECT distinct Paternita.Padre
FROM Paternita, Persone
WHERE Paternita.Figlio = Persone.Nome
AND Reddito > 20
```

Self-JOIN

- Nel JOIN tra una tabella e se stessa occorre necessariamente utilizzare dei sinonimi (alias) per distinguere le diverse occorrenze della tabella
- Query: “Le persone che guadagnano più dei rispettivi padri. Mostrare nome, reddito e reddito del padre”

```
SELECT F.Nome, F.Reddito, P.Reddito
FROM Paternita, Persone F, Persone P
WHERE
Figlio = F.Nome
AND P.Nome = Padre
AND F.Reddito > P.Reddito
```

Stessa cosa, con ridenominazione del risultato

- Query: “Le persone che guadagnano più dei rispettivi padri. Mostrare nome, reddito e reddito del padre”

```
SELECT Figlio,
       F.Reddito AS Reddito,
       P.Reddito AS RedditoPadre,
FROM Paternita, Persone P, Persone F
WHERE Figlio = F.Nome AND P.Nome = Padre
      AND F.Reddito > P.Reddito
```

Operatori aggregati

DB-Impiegati

Nome	*Cognome*	*Dipart*	*Ufficio*	*Stipendio*
Mario	Rossi	Amministrazione	10	45
Carlo	Bianchi	Produzione	20	36
Giuseppe	Verdi	Amministrazione	20	40
Franco	Neri	Distribuzione	16	45
Carlo	Rossi	Direzione	14	80
Lorenzo	Lanzi	Direzione	7	73
Paola	Borroni	Amministrazione	75	40
Marco	Franco	Produzione	20	46

Impiegato

Nome	*Indirizzo*	*Citta*
Amministrazione	Via Tito Livio	Milano
Produzione	Piazza Lavater	Torino
Distribuzione	Via Segre	Roma
Direzione	Via Tito Livio	Milano
Ricerca	Via Morone	Milano

Dipartimento

Necessità di operatori su tuple

- Nelle interrogazioni viste finora le condizioni di selezione (clausola Where) venivano valutate su *ciascuna* riga *indipendentemente* da tutte le altre

- Si può ad esempio verificare quali dipartimenti hanno sede a Milano

- Ma non si può contarne il numero, perchè occorrerebbe valutare un insieme di righe

Nome	***Indirizzo***	***Citta***
Amministrazione	Via Tito Livio	Milano
Produzione	Piazza Lavater	Torino
Distribuzione	Via Segre	Roma
Direzione	Via Tito Livio	Milano
Ricerca	Via Morone	Milano

Esempio di operatore aggregato: count

```
SELECT count(*) AS DipMilanesi
FROM Dipartimento
WHERE Citta = 'Milano'
```

DipMilanesi
3

Valutazione di un operatore aggregato

- Vediamo come viene valutata la seguente interrogazione con operatore aggregato COUNT, che conta il numero di impiegati che lavorano in Produzione

```
SELECT count(*) AS numeroImpiegati
FROM Impiegato
WHERE Dipart = ‘Produzione’
```

Valutazione di un operatore aggragato (1)

- Prima si valuta la query senza operatore aggragato

```
SELECT *
FROM Impiegato
WHERE Dipart = 'Produzione'
```

Nome	*Cognome*	*Dipart*	*Ufficio*	*Stipendio*
Carlo	Bianchi	Produzione	20	36
Marco	Franco	Produzione	20	46

Valutazione di un operatore aggragato (2)

- Poi si considerano le tuple come un insieme

```
SELECT count(*) AS numeroImpiegati
FROM Impiegato
WHERE Dipart = 'Produzione'
```

Nome	*Cognome*	*Dipart*	*Ufficio*	*Stipendio*
Carlo	Bianchi	Produzione	20	36
Marco	Franco	Produzione	20	46

- N.B.: Count conta il numero di righe

L’operatore COUNT

- COUNT può anche riferirsi a singole colonne

```
SELECT count(*) AS numeroImpiegati
FROM Impiegato
```

numeroImpiegati
8

```
SELECT count(Stipendio) AS numeroStipendi
FROM Impiegato
```

numeroStipendi
8

L'operatore COUNT

- La valutazione si effettua esattamente allo stesso modo: prima la query senza COUNT...

```
SELECT Stipendio
FROM Impiegato
```

Stipendio
45
36
40
45
80
73
40
46

L’operatore COUNT

- ... quindi il conteggio dell’insieme di righe

```
SELECT count(Stipendio) AS numeroStipendi
FROM Impiegato
```

Stipendio
45
36
40
45
80
73
40
46

numeroStipendi
8

COUNT e valori nulli

- Quando si specificano le colonne su cui contare, il risultato può variare per via dei valori nulli

- Consiferiamo la seguente tabella:

ImpiegatoConNulli

Nome	*Cognome*	*Dipart*	*Ufficio*	*Stipendio*
Mario	Rossi	Amministrazione	10	45
Carlo	Bianchi	Produzione	20	36
Giuseppe	Verdi	Amministrazione	20	40
Franco	Neri	Distribuzione	16	45
Carlo	Rossi	Direzione	14	80
Lorenzo	Lanzi	Direzione	7	*null*

COUNT e valori nulli

```
SELECT count(*) AS numeroImpiegati
FROM ImpiegatoConNulli
```

numeroImpiegati
6

```
SELECT count(Stipendio) AS numeroStipendi
FROM ImpiegatoConNulli
```

numeroStipendi
5

Conteggio delle righe diverse tra loro

- Se si vogliono considerare solo righe diverse l'una dall'altra, si può utilizzare l'opzione distinct

```
SELECT count(Stipendio) AS numeroStipendi
FROM Impiegato
```

numeroStipendi
8

```
SELECT count(distinct Stipendio) AS stipendiDiversi
FROM Impiegato
```

stipendiDiversi
6

Conteggio delle righe diverse tra loro

- Questo equivale (come al solito) alla valutazione della query senza operatore aggregato...

```
SELECT distinct Stipendio
FROM Impiegato
```

Stipendio
45
36
40
80
73
46

Conteggio delle righe diverse tra loro

- Questo equivale (come al solito) alla valutazione della query senza operatore aggregato...

```
SELECT distinct Stipendio
FROM Impiegato
```

Stipendio
45
36
40
80
73
46

- ... e al successivo conteggio delle righe

```
SELECT count (distinct Stipendio)
FROM Impiegato
```

Altri operatori

- Quanto detto per COUNT vale anche per gli operatori: SUM, MAX, MIN, AVG

- Questi operatori escludono opportunamente i valori nulli

- L’opzione distinct può ancora essere utilizzata

- Esistono altri operatori (varianza, mediano ...), ma non sono standard. Controllare il manuale del sistema che si vuole utilizzare

Esempi di altri operatori

```
SELECT max(Stipendio) AS stipendioMax
FROM Impiegato
```

stipendioMax
80

```
SELECT min(Stipendio) AS stipendioMin
FROM Impiegato
```

stipendioMin
36

Altri operatori

```
SELECT sum(Stipendio) AS sommaStipendi
FROM Impiegato
```

sommaStipendi
405

```
SELECT avg(Stipendio) AS mediaStipendi
FROM Impiegato
```

mediaStipendi
50.625

Operatori aggregati e JOIN

- Gli operatori aggregati si possono utilizzare anche in concomitanza con i JOIN

```
SELECT max(Stipendio) AS stipendioMassimo
FROM Impiegato, Dipartimento D
where Dipart = D.Nome AND Citta = ‘Milano’
```

stipendioMassimo
80

Operatori aggregati e ridenominazione

- Se non utilizziamo la AS, il risultato non ha nome

```
SELECT max(Stipendio)
FROM Impiegato, Dipartimento D
where Dipart = D.Nome AND Citta = ‘Milano’
```

max(Stipendio)
80

Operatori aggregati e target list

- Non è lecita la presenza contemporanea nella target list di nomi di campi e operatori aggregati

- Ad esempio, la seguente interrogazione non è corretta:

```
SELECT Cognome, Nome, min(Stipendio)
FROM Impiegato
where Dipart = ‘Amministrazione’
```

Interrogazioni con raggruppamento

- Gli operatori aggregati vengono applicati ad un insieme di righe

- Gli esempi visti valutavano gli operatori su tutte le righe di una tabella

- Spesso esiste l'esigenza di applicare operatori aggregati distintamente a particolari sottoinsiemi delle righe di una tabella

- Ad esempio: *per ogni dipartimento*, trovare la somma degli stipendi

Esempio di raggruppamento (1)

- Query: “Per ogni dipartimento, la somma degli stipendi”
- Intuitivamente, occorre selezionare in principio le informazioni di interesse, ovvero il dipartimento e gli stipendi

Dipart	*Stipendio*
Amministrazione	45
Produzione	36
Amministrazione	40
Distribuzione	45
Direzione	80
Direzione	73
Amministrazione	40
Produzione	46

Esempio di raggruppamento (2)

- Query: “Per ogni dipartimento, la somma degli stipendi”
- Poi raggruppiamo per il dipartimento

Dipart	*Stipendio*
Amministrazione	45
Amministrazione	40
Amministrazione	40
Produzione	36
Produzione	46
Distribuzione	45
Direzione	80
Direzione	73

Esempio di raggruppamento (3)

- Query: “Per ogni dipartimento, la somma degli stipendi”
- Infine calcoliamo la somma degli stipendi

Dipart	*TotaleStipendi*
Amministrazione	125
Produzione	82
Distribuzione	45
Direzione	153

Esempio di raggruppamento (SQL)

- Query: “Per ogni dipartimento, la somma degli stipendi”

```
SELECT Dipart, sum(Stipendio) as TotaleStipendi
FROM Impiegato
GROUP BY Dipart
```

Dipart	*TotaleStipendi*
Amministrazione	125
Produzione	82
Distribuzione	45
Direzione	153

Valutazione di query con raggruppamento (1)

- Query: “Per ogni dipartimento, la somma degli stipendi”

```
SELECT Dipart, Stipendio
FROM Impiegato
```

Dipart	*Stipendio*
Amministrazione	45
Produzione	36
Amministrazione	40
Distribuzione	45
Direzione	80
Direzione	73
Amministrazione	40
Produzione	46

Valutazione di query con raggruppamento (2)

- Query: "Per ogni dipartimento, la somma degli stipendi"

```
SELECT Dipart, Stipendio
FROM Impiegato
GROUP BY Dipart
```

Dipart	*Stipendio*
Amministrazione	45
Amministrazione	40
Amministrazione	40
Produzione	36
Produzione	46
Distribuzione	45
Direzione	80
Direzione	73

Valutazione di query con raggruppamento (3)

- Query: “Per ogni dipartimento, la somma degli stipendi”

```
SELECT Dipart, sum(Stipendio) AS TotaleStipendi
FROM Impiegato
GROUP BY Dipart
```

Dipart	*TotaleStipendi*
Amministrazione	125
Produzione	82
Distribuzione	45
Direzione	153

Operatori aggregati e target list

- ATTENZIONE: nel momento i cui si utilizzano operatori aggregati, si stanno considerando insiemi di righe, non singole righe

- Di conseguenza, non è possibile utilizzare nelle target list attributi non utilizzati per il raggruppamento

- Infatti, questi attributi possono presentare più valori per ogni insieme di tuple. Non è quindi possibile ottenere un singolo valore per ogni gruppo di righe

Operatori aggregati e target list

- Ad esempio, la seguente interrogazione NON HA SENSO

```
SELECT Cognome, sum(Stipendio) AS TotaleStipendi
FROM Impiegato
GROUP BY Dipart
```

Cognome	**Dipart**	**TotaleStipendi**
Rossi Verdi Borroni	Amministrazione	125
Bianchi Franco	Produzione	82
Neri	Distribuzione	45
Rossi Lanzi	Direzione	153

Operatori aggregati e target list

- Ad esempio, la seguente interrogazione NON HA SENSO

```
SELECT Cognome, sum(Stipendio) AS TotaleStipendi
FROM Impiegato
GROUP BY Dipart
```

Quale cognome dovremmo scegliere?

Cognome	**Dipart**	**TotaleStipendi**
Rossi Verdi Borroni	Amministrazione	125
Bianchi Franco	Produzione	82
Neri	Distribuzione	45
Rossi Lanzi	Direzione	153

Operatori aggregati e target list

- Le interrogazioni che abbiamo visto precedentemente, con funzioni aggregate e senza GROUP BY, possono essere pensate come query in cui il GROUP BY produce un solo insieme di righe

- Continua dunque a valere la regola di non utilizzare attributi nella target list, se essi non sono stati usati per il raggruppamento

- Poichè in assenza del GROUP BY nessun attributo viene utilizzato per il raggruppamento, se si utilizzano funzioni aggregate non si possono specificare altri attributi nella target list

Operatori aggregati e target list

- Query corretta:

```
SELECT min(Stipendio), max(Stipendio)
FROM Impiegato
```

- Query NON corretta:

```
SELECT Cognome, max(Stipendio)
FROM Impiegato
```

Condizioni sui gruppi

- Ovviamente, anche utilizzando GROUP BY è possibile filtrare le righe sulla base di predicati

- Ad esempio:

```
SELECT min(Stipendio), max(Stipendio)
FROM Impiegato
WHERE Ufficio = 20
GROUP BY Dipart
```

Condizioni sui gruppi

- Se le condizioni sono però da calcolare sui raggruppamenti di tuple, si utilizza la clausola HAVING

- Ciò accade quando le condizioni utilizzano funzioni aggregate

```
SELECT Dipart, sum(Stipendio)
FROM Impiegato
GROUP BY Dipart
HAVING sum(Stipendio) > 100
```

Dipart	
Amministrazione	125
Direzione	153

WHERE o HAVING?

- Per decidere se specificare le condizioni nella clausola WHERE o tramite HAVING, la regola è semplice:

  ➡ Se bisogna utilizzare una funzione aggregata, significa che la condizione concerne gli insiemi di tuple: HAVING

  ➡ In caso contrario: WHERE

WHERE o HAVING?

- “I diparimenti per cui la media degli stipendi degli impiegati che lavorano nell’ufficio 20 è superiore a 25 milioni”

```
SELECT Dipart
FROM Impiegato
WHERE Ufficio = 20
GROUP BY Dipart
HAVING avg(Stipendio) > 25
```

Riassumiamo

- SQL:

  SELECT Lista_Attributi_o_Espressioni

  FROM Lista_Tabelle

  [WHERE Condizioni_Semplici]

  [GROUP BY Lista_Attributi_Di_Raggruppamento]

  [HAVING Condizioni_Aggregate]

  [ORDER BY Lista_Attributi_Di_Ordinamento]