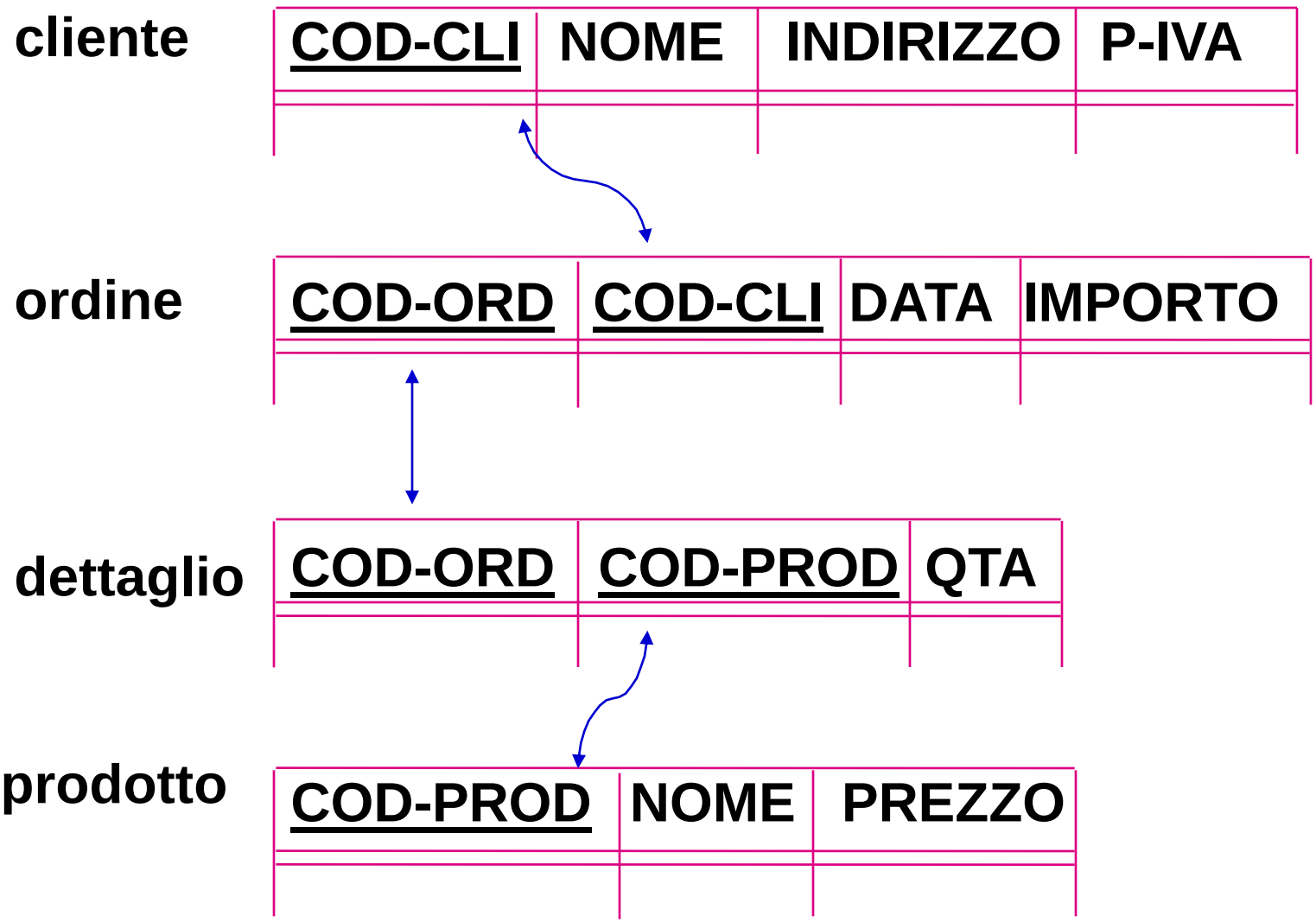
Interrogazioni complesse

Classificazione delle interrogazioni complesse

- **Query con ordinamento**
- **Query con aggregazione**
- **Query con raggruppamento**
- **Query binarie**
- **Query annidate**

Esempio : gestione ordini



Istanza di ordine

ordine

COD-ORD	COD-CLI	DATA	IMPORTO
1	3	1997-06-01	50.000
2	4	1997-08-03	8.000
3	3	1997-09-01	5.500
4	1	1997-07-01	12.000
5	1	1997-08-01	1.500
6	3	1997-09-03	27.000

Query con ordinamento

SELECT * FROM ORDINE
WHERE IMPORTO > 100 ORDER BY DATA

COD-ORD	COD-CLI	DATA	IMPORTO
1	3	1997-06-01	50.000
4	1	1997-07-01	12.000
5	1	1997-08-01	1.500
2	4	1997-08-03	8.000
3	3	1997-09-01	5.500
6	3	1997-09-03	27.000

Order by

```
SELECT  * FROM ORDINE
WHERE IMPORTO > 100
ORDER BY COD-CLI
```

COD-ORD	COD-CLI	DATA	IMPORTO
4	1	1997-07-01	12.000
5	1	1997-08-01	1.500
1	3	1997-06-01	50.000
6	3	1997-09-03	5.500
3	3	1997-09-01	27.000
2	4	1997-08-03	8.000

Order by

Nel caso si voglia ordinare sui valori di una colonna risultato di una espressione si usa una notazione "posizionale"

```
SELECT COD-CLI, IMPORTO/1.000
FROM ORDINE
WHERE IMPORTO > 10.000
ORDER BY 2
```

(ordina sulla SECONDA colonna)

Order by

.....

ORDER BY COD-CLI ASC, DATA DESC

COD-ORD	COD-CLI	DATA	IMPORTO
5	1	1997-08-01	1.500
4	1	1997-07-01	12.000
6	3	1997-09-03	27.000
3	3	1997-09-01	5.500
1	3	1997-06-01	50.000
2	4	1997-08-03	8.000

Query aggregate

Utilizzano le funzioni aggregate:

SUM **sommatoria**

AVG **media**

MIN **minimo**

MAX **massimo**

COUNT **cardinalità**

(I NULL sono esclusi)

Query con massimo

- **Selezionare l'importo massimo degli ordini.**

```
SELECT MAX(IMPORTO) AS MAX-IMP
FROM ORDINE
```

MAX-IMP
50.000

Query con sommatoria

- **Selezionare la somma degli importi degli ordini relativi al cliente numero 1.**

```
SELECT SUM(IMPORTO) AS SOMMA-IMP
FROM ORDINE
WHERE COD-CLIENTE = 1
```

SOMMA-IMP
13.500

Query con raggruppamento

si aggiungono le clausole

GROUP-BY (raggruppamento)

HAVING (selezione dei gruppi)

```
SELECT  …..
FROM …..
WHERE …..
GROUP BY …..
HAVING …..
```

Query con raggruppamento

- **Selezionare la somma degli importi degli ordini successivi al 10-6-97 per quei clienti che hanno emesso almeno 2 ordini.**

```
SELECT  COD-CLI,  SUM (IMPORTO)
FROM ORDINE
WHERE DATA > '1997-06-10'
GROUP BY COD-CLI
HAVING COUNT (IMPORTO)  >= 2
```

Passo 1: Valutazione WHERE

COD-ORD	COD-CLI	DATA	IMPORTO
2	4	1997-08-03	8.000
3	3	1997-09-01	5.500
4	1	1997-07-01	12.000
5	1	1997-08-01	1.500
6	3	1997-09-03	27.000

(WHERE DATA > '1997-06-10')

Passo 2 : Raggruppamento

si valuta la clausola GROUP-BY

COD-ORD	COD-CLI	DATA	IMPORTO
4	1	1997-07-01	12.000
5	1	1997-08-01	1.500
3	3	1997-09-01	5.500
6	3	1997-09-03	27.000
2	4	1997-08-03	8.000

(GROUP BY COD-CLI)

Passo 3 : Calcolo degli aggregati

per ciascun gruppo si calcolano
SUM (IMPORTO) e COUNT (IMPORTO)

COD-CLI	SUM (IMPORTO)	COUNT (IMPORTO)
1	13.500	2
3	32.500	2
4	8.000	1

Passo 4 : Selezione dei gruppi

si valuta il predicato

COUNT (IMPORTO) >=2

COD-CLI	SUM (IMPORTO)	COUNT (IMPORTO)
1	13.500	2
3	32.500	2
~~4~~	~~8.000~~	~~1~~

Passo 5 : Produzione del risultato

COD-CLI	SUM (IMPORTO)
1	13.500
3	32.500

Doppio raggruppamento

- **Selezionare la somma delle quantità dei dettagli degli ordini emessi da ciascun cliente per ciascun prodotto, purché la somma superi 50.**

```
SELECT COD-CLI, COD-PROD, SUM(QTA)
FROM ORDINE AS O, DETTAGLIO AS D
WHERE  O.COD-ORD = D.COD-ORD
GROUP BY COD-CLI, COD-PROD
HAVING SUM(QTA) > 50
```

Situazione dopo il join e il raggruppamento

ordine		dettaglio		
COD-CLI	ORDINE. COD-ORD	DETTAGLIO. COD-ORD	COD-PROD	QTA
1	3	3	1	30
1	4	4	1	20
1	3	3	2	30
1	5	5	2	10
2	3	3	1	60
3	1	1	1	40
3	2	2	1	30
3	6	6	1	25

gruppo 1,1

gruppo 1,2

gruppo 2,1

gruppo 3,1

Estrazione del risultato

si valuta la funzione aggregata

SUM(QTA) e il predicato HAVING

COD-CLI	COD-PROD	SUM(QTA)
~~1~~	~~1~~	~~50~~
~~1~~	~~2~~	~~40~~
2	1	60
3	1	95

Query con raggruppamento e ordinamento

- **E’ possibile ordinare il risultato anche delle query con raggruppamento**

```
SELECT  …..
FROM …..
[ WHERE ….. ]
GROUP BY …..
[ HAVING ….. ]
ORDER BY ….
```

Raggruppamento e ordinamento

- **Selezionare la somma degli importi degli ordini successivi al 10/6/97 per quei clienti che hanno emesso almeno 2 ordini dopo quella data, in ordine di cliente.**

```
SELECT  COD-CLI,  SUM (IMPORTO)
FROM ORDINE
WHERE DATA > '1997-06-10'
GROUP BY COD-CLI
HAVING COUNT (IMPORTO)  >= 2
ORDER BY COD-CLI
```

ESERCIZI SUL GROUP BY

1
```
SELECT  SUM(NUM)
FROM SPEDIZIONI
GROUP BY ART-NO
```

OPPURE

2
```
SELECT  SUM(NUM)
FROM SPEDIZIONI
GROUP BY ART-NO, SPED-NO
```

ESERCIZI SUL GROUP BY

SU:

ART-NO	SPED-NO	NUM	(SPEDIZIONI)
1	2	100	
1	2	25	
1	3	75	
2	2	100	
2	4	100	
2	4	75	
1	4	25	

ESERCIZI SUL GROUP BY

1 ….GROUP BY ART-NO

ART-NO	SPED-NO	NUM
1	2,2,3,4	100,25,75,25
2	2,4,4	100,100,75

1 ….GROUP BY ART-NO, SPED-NO

ART-NO	SPED-NO	NUM
1	2	100,25
1	3	75
1	4	25
2	2	100
2	4	100,75

ESERCIZI SUL GROUP BY

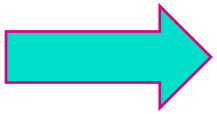
SELECT ARTICOLO

FROM VENDITE GROUP BY ARTICOLO

HAVING COUNT(DISTINCT DATA) > 1

SU

ARTICOLO	DATA	NUM
PENNA	1997-12-13	2
MATITA	1997-12-08	1
GOMMA	1997-12-10	1
MATITA	1997-12-08	2
PENNA	1997-12-08	1



ESERCIZI SUL GROUP BY

GROUP BY ARTICOLO

SU

ARTICOLO	DATA
PENNA	1997-12-13, 1997-12-08
MATITA	1997-12-08, 1997-12-08
GOMMA	1997-12-10

HAVING COUNT(DISTINCT DATA) > 1

ARTICOLO
PENNA

Query binarie

Costruite concatenando due query SQL tramite operatori insiemistici:

UNION	**unione**
INTERSECT	**intersezione**
EXCEPT (MINUS)	**differenza**

(si eliminano i duplicati)

Unione

- **Selezionare i codici degli ordini i cui importi superano € 5.000 oppure presenti in qualche dettaglio con quantità superiore a 1000.**

```
SELECT COD-ORD
FROM ORDINE
WHERE IMPORTO > 5.000
    UNION
SELECT COD-ORD
FROM DETTAGLIO
WHERE QTA > 1000
```

Differenza

- **Selezionare i codici degli ordini i cui importi superano € 5.000 ma non presenti in nessun dettaglio con quantità superiore a 1000.**

```
SELECT COD-ORD
FROM ORDINE
WHERE IMPORTO > 5.000
    EXCEPT
SELECT COD-ORD
FROM DETTAGLIO
WHERE QTA > 1000
```

Intersezione

- **Selezionare i codici degli ordini i cui importi superano € 5.000 e che sono presenti in qualche dettaglio con quantità superiore a 1000.**

```
SELECT COD-ORD
FROM ORDINE
WHERE IMPORTO > 5.000
    INTERSECT
SELECT COD-ORD
FROM DETTAGLIO
WHERE QTA > 1000
```

Query annidate

Costruite concatenando due query SQL nel predicato where:

[NOT] IN	**appartenenza**
[DOES NOT] EXISTS	**esistenza**
ANY, ALL	**quantificatori**

comparatore: =, !=, <, <=, >, >=

Query annidate (nested)

- **Selezionare nome e indirizzo dei clienti che hanno emesso qualche ordine di importo superiore a € 10.000.**

```
SELECT NOME, INDIRIZZO
FROM CLIENTE
WHERE COD-CLI IN
        ( SELECT COD-CLI
          FROM ORDINE
          WHERE IMPORTO > 10.000 )
```

Equivalenza fra IN e query semplici

```
SELECT NOME, INDIRIZZO
FROM CLIENTE
WHERE COD-CLI IN
      ( SELECT COD-CLI
        FROM ORDINE
        WHERE IMPORTO > 10.000 )
```

equivale (a meno di duplicati) a:

```
SELECT NOME, INDIRIZZO
FROM CLIENTE JOIN ORDINE
      ON CLIENTE.COD-CLI = ORDINE.COD-CLI
WHERE IMPORTO > 10.000
```

Nested Query complesse (fino a 8 livelli)

- **Selezionare nome e indirizzo dei clienti che hanno emesso qualche ordine i cui dettagli comprendono il prodotto “Pneumatico”.**

```
SELECT NOME, INDIRIZZO  FROM CLIENTE
WHERE COD-CLI IN
  ( SELECT COD-CLI  FROM ORDINE
    WHERE COD-ORD IN
      ( SELECT COD-ORD  FROM DETTAGLIO
        WHERE COD-PROD IN
          ( SELECT COD-PROD  FROM PRODOTTO
            WHERE NOME = 'Pneumatico' ) ) )
```

La query equivalente

equivale (a meno di duplicati) a:

```
SELECT NOME, INDIRIZZO
FROM CLIENTE AS C, ORDINE AS O,
  DETTAGLIO AS D, PRODOTTO AS P
WHERE C.COD-CLI = O.COD-CLI
  AND O.COD-ORD = D.COD-ORD
  AND D.COD-PROD = P.COD-PROD
  AND NOME= 'Pneumatico'
```

Nested query correlate (con variabile)

```
SELECT NOME
FROM IMPIEGATI AS X,
 WHERE SALARIO>
   ( SELECT SALARIO
     FROM IMPIEGATI
     WHERE MATRICOLA= X.MANAGER )
```

sulla relazione IMPIEGATI (matricola, nome, manager…)
equivale al JOIN

```
SELECT X.NOME
FROM IMPIEGATI AS X, IMPIEGATI AS Y
WHERE X.SALARIO > Y.SALARIO
  AND Y.MATRICOLA= X.MANAGER
```

Uso di IN nelle modifiche

- **aumentare di € 50 l'importo di tutti gli ordini che comprendono il prodotto 456**

```
UPDATE ORDINE
SET IMPORTO = IMPORTO + 50
WHERE COD-ORD IN
      ( SELECT COD-ORD
        FROM DETTAGLIO
        WHERE COD-PROD = '456' )
```

Query con NOT IN

- **Selezionare nome e indirizzo dei clienti che non hanno emesso nessun ordine di importo superiore a € 10.000.**

```
SELECT NOME, INDIRIZZO
FROM CLIENTE
WHERE COD-CLI NOT IN
        ( SELECT COD-CLI
          FROM ORDINE
          WHERE IMPORTO > 10.000 )
```

La query con ANY e ALL

```
SELECT COD-ORD
FROM ORDINE
WHERE IMPORTO > ANY
      ( SELECT IMPORTO
        FROM ORDINE )
```

```
SELECT COD-ORD
FROM ORDINE
WHERE IMPORTO >= ALL
      ( SELECT IMPORTO
        FROM ORDINE )
```

COD-ORD	IMPORTO
1	5.000
2	30.000
3	9.000

ANY	ALL
F	F
V	V
V	F

Esercizi

- **Riprendere le basi di dati per la gestione del personale e degli ordini ed esprimere in SQL le interrogazioni :**
  - **quale impiegati lavorano in un progetto in cui non lavora il loro manager?**
  - **quanti ordini ha emesso Paolo?**
  - **quante candele sono state ordinate il 5/7/97?**
  - **calcolare per ciascun cliente la somma degli importi di tutti gli ordini**
  - **estrarre l'ordine di importo più alto**

Esercizi (lezione precedente)

- **in quali tipi di progetti lavora Giovanni?**

```
SELECT TIPO  FROM PROGETTO
WHERE NUM-PROG IN
  ( SELECT NUM-PROG FROM ASSEGNAMENTO
     WHERE MATR IN
        ( SELECT MATR FROM IMPIEGATO
          WHERE NOME='Giovanni' ) )
```

- **chi è il manager di Piero?**

```
SELECT NOME  FROM IMPIEGATO
WHERE MATR IN
       ( SELECT MATR-MGR FROM IMPIEGATO
         WHERE NOME='Piero' )
```

Query annidate (nested)

- **Selezionare nome e indirizzo dei clienti che hanno emesso qualche**
  [o non hanno emesso nessun]
  ordine di importo superiore a € 10.000

```
SELECT NOME, INDIRIZZO
FROM CLIENTE C
WHERE [NOT] EXISTS
      (   SELECT *
          FROM ORDINE
          WHERE IMPORTO > 10.000
            AND COD_CLI=C.COD_CLI       )
```

Query universali

- **Esempio:**
  Trovare gli ordini che contengono tutti i prodotti

- **In Algebra relazionale queste interrogazioni si possono esprimere tramite l'operatore di divisione ÷**

- **Ma in SQL??**
- **Non c'è un quantificatore universale (per ogni) ma solo un quantificatore esistenziale (EXISTS)**

Query universali

- **In generale va ribaltata la logica della query, per esprimere la quantificazione universale per mezzo dell'operatore EXISTS e una doppia negazione**

Anche in logica solo uno dei due quantificatori è in realtà necessario e l'altro può essere espresso per mezzo del primo:

$$\forall x\ P(x) \equiv \neg\exists x\ \neg P(x)$$

Per ogni x vale la proprietà P equivale a dire:
Non esiste alcun x per cui non vale la proprietà P

Query universali

- **Selezionare gli ordini che contengono tutti i prodotti (= Selezionare gli ordini per cui non esiste alcun prodotto che non sia compreso nelle loro righe di dettaglio)**

```
SELECT * FROM ORDINE ORD
WHERE NOT EXISTS
   (SELECT * FROM PRODOTTO
    WHERE COD-PROD NOT IN
       (SELECT COD-PROD FROM DETTAGLIO DET
        WHERE DET.COD-ORD=ORD.COD-ORD ) )
```

Query universali

- **Selezionare gli ordini che contengono tutti i prodotti (= Selezionare gli ordini per cui non esiste alcun prodotto per cui non esiste alcuna loro riga di dettaglio che lo contiene)**

```
SELECT * FROM ORDINE ORD
WHERE NOT EXISTS
   (SELECT * FROM PRODOTTO PRO
    WHERE NOT EXISTS
       (SELECT * FROM DETTAGLIO DET
         WHERE DET.COD-ORD=ORD.COD-ORD
              AND DET.COD-PROD=PRO.COD-PROD ) )
```