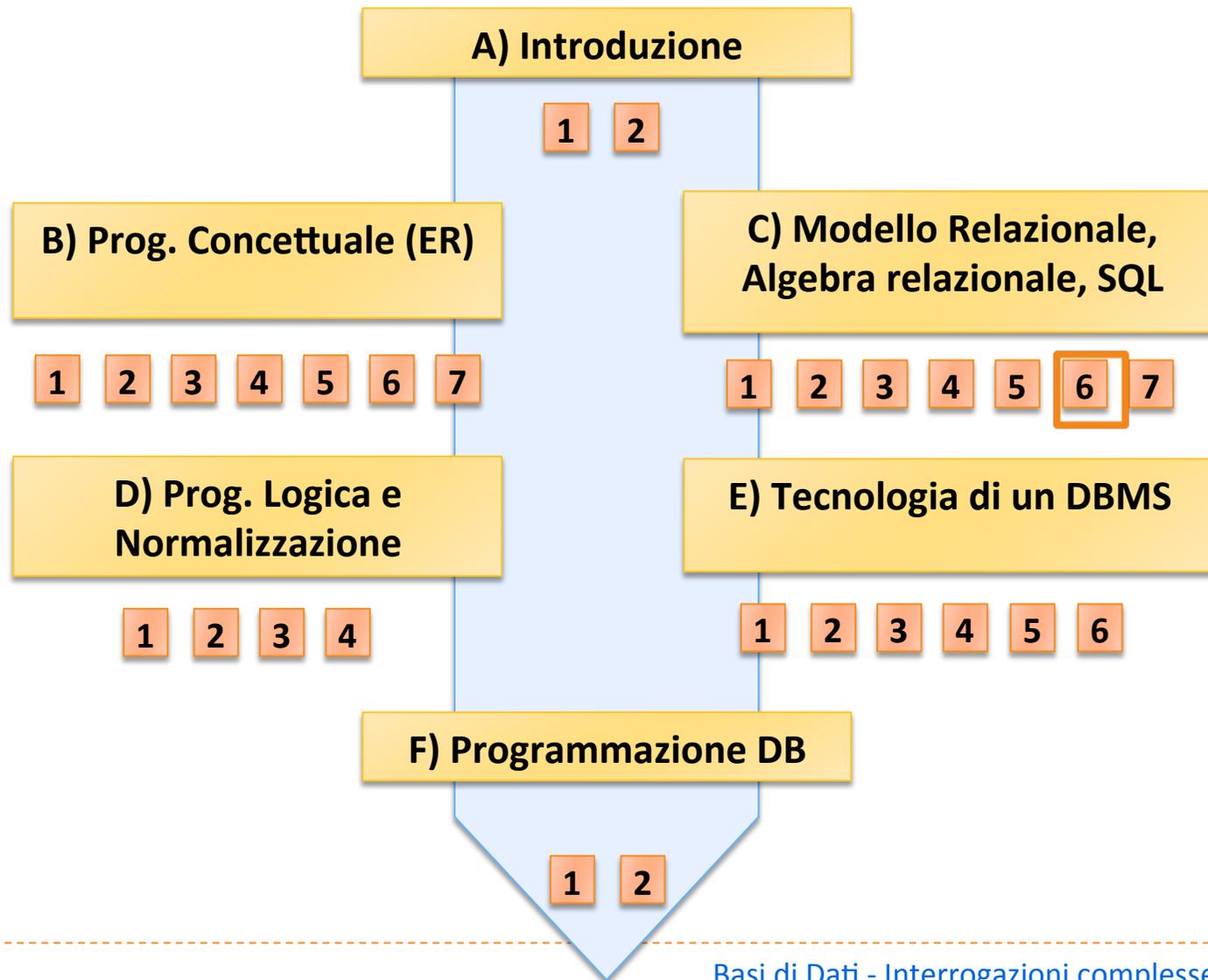


# Basi di Dati

Interrogazioni complesse (II parte)

# Basi di Dati – Dove ci troviamo?



# Classificazione delle interrogazioni complesse

---

- ▶ Query con ordinamento
- ▶ Query con aggregazioni
- ▶ Query con raggruppamento
- ▶ Query binarie
- ▶ Query nidificate

## Esempio : gestione ordini

---

**cliente**

<u>COD-CLI</u>	NOME	INDIRIZZO	P-IVA

**ordine**

<u>COD-ORD</u>	<u>COD-CLI</u>	DATA	IMPORTO

**dettaglio**

<u>COD-ORD</u>	<u>COD-PROD</u>	QTA

**prodotto**

<u>COD-PROD</u>	NOME	PREZZO

# Istanza di ordine

---

## ordine

<u>COD-ORD</u>	<u>COD-CLI</u>	DATA	IMPORTO
1	3	2004-06-01	50.000
2	4	2004-08-03	8.000
3	3	2004-09-01	5.500
4	1	2004-07-01	12.000
5	1	2004-08-01	1.500
6	3	2004-09-03	27.000

## Query binarie

---

- ▶ Costruite concatenando due query SQL tramite operatori insiemistici:
  - ▶ UNION → unione
  - ▶ INTERSECT → intersezione
  - ▶ EXCEPT → differenza
  
- ▶ si eliminano i duplicati

## Unione

---

- ▶ Selezionare i codici degli ordini i cui importi superano 500 € oppure presenti in qualche dettaglio con quantità superiore a 1000.

```
SELECT COD-ORD
FROM ORDINE
WHERE IMPORTO > 500
      UNION
SELECT COD-ORD
FROM DETTAGLIO
WHERE QTA > 1000
```

## Differenza

---

- ▶ Selezionare i codici degli ordini i cui importi superano 500 € **ma non** presenti in nessun dettaglio con quantità superiore a 1000.

```
SELECT COD-ORD  
FROM ORDINE  
WHERE IMPORTO > 500
```

**EXCEPT**

```
SELECT COD-ORD  
FROM DETTAGLIO  
WHERE QTA > 1000
```

## Intersezione

---

- ▶ Selezionare i codici degli ordini i cui importi superano 500 € e che sono presenti in qualche dettaglio con quantità superiore a 1000.

```
SELECT COD-ORD  
FROM ORDINE  
WHERE IMPORTO > 500
```

**INTERSECT**

```
SELECT COD-ORD  
FROM DETTAGLIO  
WHERE QTA > 1000
```

## Query nidificate

---

- ▶ Costruite concatenando due query SQL nel predicato where:
  - ▶ [NOT] IN → appartenenza
  - ▶ [NOT] EXISTS → esistenza
  - ▶ ANY, ALL → quantificatori
  
- ▶ comparatore: =, !=, <, <=, >, >=

## Query con ANY e ALL

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

COD-ORD	IMPORTO
1	50
2	300
3	90

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

ANY	ALL
F	F
V	V
V	F

## Query con EXISTS

---

- ▶ Il predicato **EXISTS (<subquery>)** ha valore true se e solo se l'insieme di valori restituiti da <subquery> è non vuoto.
- ▶ Selezionare nome e indirizzo dei clienti che hanno emesso qualche ordine di importo superiore a 10.000 €.

```
SELECT NOME, INDIRIZZO  
FROM CLIENTE C  
WHERE EXISTS  
    (SELECT *  
     FROM ORDINE O  
     WHERE C.CODCLI=O.CODCLI  
     AND IMPORTO>10.000)
```

## Query con NOT EXISTS

---

- ▶ Il predicato **NOT EXISTS (<subquery>)** ha valore true se e solo se l'insieme di valori restituiti da <subquery> è vuoto.
- ▶ Selezionare nome e indirizzo dei clienti che non hanno emesso alcun ordine di importo superiore a 10.000 €.

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

## Query con IN

---

- ▶ 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)
```

## 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)
```

## 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, ORDINE
WHERE CLIENTE.COD-CLI = ORDINE.COD-CLI
AND IMPORTO > 10.000
```

## Nested Query complesse

---

- ▶ 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'
```

## Uso di IN nelle modifiche

---

- ▶ aumentare di 5 € l'importo di tutti gli ordini che comprendono il prodotto 456

```
UPDATE ORDINE
SET IMPORTO = IMPORTO + 5
WHERE COD-ORD IN
    (SELECT COD-ORD
     FROM DETTAGLIO
     WHERE COD-PROD = '456')
```

# Esempio : gestione personale

---

## impiegato

MATR	NOME	DATA-ASS	SALARIO	MATR-MGR
1	Piero	1-1-02	1500 €	2
2	Giorgio	1-1-04	2000 €	null
3	Giovanni	1-7-03	1000 €	2

## assegnamento

MATR	NUM-PROG	PERC
1	3	50
1	4	50
2	3	100
3	4	100

## progetto

NUM-PROG	TITOLO	TIPO
3	Idea	Esprit
4	Wide	Esprit

## Nested query correlate (con variabile)

---

- ▶ Selezionare il nome degli impiegati il cui salario è maggiore di quello del proprio manager.

```
SELECT NOME
FROM IMPIEGATI AS X,
WHERE SALARIO >
(
  SELECT SALARIO
  FROM IMPIEGATI
  WHERE MATR = X.MATR-MGR
)
```

## Nested query correlate (con variabile)

---

```
SELECT NOME
FROM IMPIEGATI AS X,
WHERE SALARIO >
  ( SELECT SALARIO
    FROM IMPIEGATI
    WHERE MATR = X.MATR-MGR )
```

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.MATR = X.MATR-MGR
```

## Riduzione di query innestate

---

- ▶ Le query innestate formulate con i seguenti operatori si possono ridurre a query semplici equivalenti (stessa risposta per ogni possibile istanza della base di dati):
  - ▶ IN
  - ▶ ANY (con qualsiasi operatore di confronto)
  - ▶ EXISTS con subquery correlata

## Riduzione di query innestate

---

- ▶ Attenzione!
- ▶ Come si è visto, è spesso possibile ridurre una query innestata a una query semplice.
- ▶ Tuttavia, le query innestate formulate con i seguenti operatori non si possono ridurre:
  - ▶ NOT IN
  - ▶ ALL (con qualsiasi operatore di confronto)
  - ▶ NOT EXISTS con subquery correlata

## Esempi

---

- ▶ 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'))
```

## Esempi

---

- ▶ chi è il manager di Piero?

```
SELECT NOME FROM IMPIEGATO
```

```
WHERE MATR IN
```

```
(SELECT MATR-MGR FROM IMPIEGATO
```

```
WHERE NOME='Piero')
```

## Divisione in SQL

---

- ▶ L'operazione di divisione, vista in algebra relazionale, non è definita in SQL. Pertanto, le interrogazioni che richiedono tale operatore, come ad esempio:
  - ▶ Selezionare i dati degli ordini che contengono tutti i prodotti di prezzo > 100

devono essere riformulate con una doppia negazione nel seguente modo:

- ▶ Selezionare i dati degli ordini per i quali non esiste alcun prodotto di prezzo > 100 che non sia contenuto in essi

## Divisione in SQL

---

- ▶ Selezionare i dati degli ordini per i quali non esiste alcun prodotto di prezzo > 100 che non sia contenuto in essi

```
SELECT *
FROM ORDINE O
WHERE NOT EXISTS
  ( SELECT *
    FROM PRODOTTO P
    WHERE P.PREZZO > 100
      AND NOT EXISTS
        ( SELECT *
          FROM DETTAGLIO D
          WHERE D.COD-PROD = P.COD-PROD
            AND D.COD-ORD = O.COD-ORD))
```

## Divisione in SQL

---

- ▶ Come sappiamo, invece, in algebra relazionale possiamo più semplicemente sfruttare l'operatore di divisione ...

$$\text{ORDINE} \bowtie \left( \pi_{\text{COD-ORD,COD-PROD}} \text{DETTAGLIO} \div \pi_{\text{COD-PROD}} (\sigma_{\text{PREZZO} > 100} \text{PRODOTTO}) \right)$$

## 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 5/7/04?
  - ▶ calcolare per ciascun cliente la somma degli importi di tutti gli ordini
  - ▶ estrarre l'ordine di importo più alto
  - ▶ quali ordini contengono tutti i prodotti il cui nome inizia per "iPhone"?