EPICODE

Combinare più tabelle utilizzando le JOIN
Le SUBQUERY

AGENDA

- ❑ Il concetto di JOIN
- ❑ Le subquery

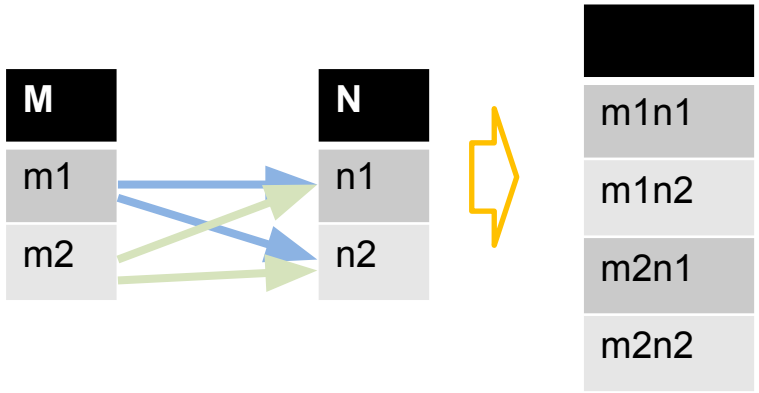
Molto spesso occorre recuperare informazioni da più tabelle. È fondamentale dunque saper combinare correttamente tra loro più tabelle.

Alla fine di questo modulo sarai in grado di:

- ✔ Scrivere query che interessano più tabelle
- ✔ Scegliere la giusta logica di join in base al risultato richiesto
- ✔ Innestare le query

SQL: .. Assodiamo alcuni aspetti

Dati due insiemi, *M* e *N*, rispettivamente di *m* e *n* valori, il prodotto cartesiano è l'insieme di tutte le possibili combinazioni cioè un insieme di *m*n* valori.



Il prodotto cartesiano, l'insieme di tutte le possibili combinazioni di valori di due insiemi, è alla base della **JOIN** tra tabelle: se la prima tabella (insieme *M*) ha 2 record (*m*=2) e la seconda tabella (insieme *N*) ha 2 record (*n*=2), il prodotto cartesiano restituisce 2∗2 = 4 record (*m*n* record).

SQL: CROSS JOIN

Il **CROSS JOIN** è il tipo di join più semplice e restituisce il prodotto cartesiano cioè le possibili combinazioni.

ProductID	Product	CategoryID
1	Adjustable Race	NULL
343	Road-650 Black, 52	2
344	Mountain-100 Silver, 38	1

CategoryID	Category
1	Mountain Bikes
2	Road Bikes
3	Touring Bikes

```
SELECT
p1.Product
, p1.CategoryID
, p2.CategoryID
, p2.Category
FROM p1 CROSS JOIN p2
```

	Product	CategoryID	CategoryID	Category
1	Adjustable Race	NULL	1	Mountain Bikes
2	Road-650 Black, 52	2	1	Mountain Bikes
3	Mountain-100 Silver, 38	1	1	Mountain Bikes
4	Adjustable Race	NULL	2	Road Bikes
5	Road-650 Black, 52	2	2	Road Bikes
6	Mountain-100 Silver, 38	1	2	Road Bikes
7	Adjustable Race	NULL	3	Touring Bikes
8	Road-650 Black, 52	2	3	Touring Bikes
9	Mountain-100 Silver, 38	1	3	Touring Bikes

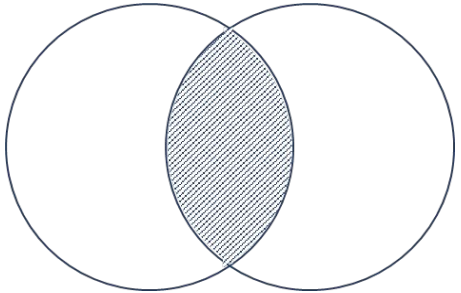
SQL: INNER JOIN

L'**INNER JOIN** consente di combinare i record di due tabelle che includono valori corrispondenti in un campo in comune.

Date le due tabelle, il comune in comune è **CategoryID** (tipicamente il campo in comune è una PK da un lato e una FK dall'altro...). L' **INNER JOIN** combina i record per cui c'è corrispondenza rispetto al campo in comune

ProductID	Product	CategoryID
1	Adjustable Race	NULL
343	Road-650 Black, 52	2
344	Mountain-100 Silver, 38	1

CategoryID	Category
1	Mountain Bikes
2	Road Bikes
3	Touring Bikes



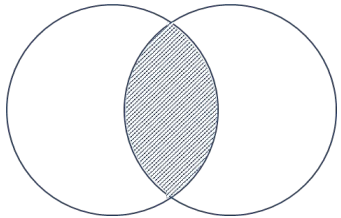
L'INNER JOIN restituisce l'intersezione degli insiemi

SQL: INNER JOIN

L'**INNER JOIN** consente di combinare i record di due tabelle che includono valori corrispondenti in un campo in comune. Restituisce l'intersezione degli insiemi.
Supponiamo di dover interrogare le tabelle Product e Category al fine di esporre i soli prodotti che hanno una categoria.

ProductID	Product	CategoryID
1	Adjustable Race	NULL
343	Road-650 Black, 52	2
344	Mountain-100 Silver, 38	1

CategoryID	Category
1	Mountain Bikes
2	Road Bikes
3	Touring Bikes



```
SELECT
P.ProductID
, P.Product
, C.Category
, C.CategoryID
, P.CategoryID
FROM Product AS P INNER JOIN
Category AS C
ON P.CategoryID = C.CategoryID
```

	ProductID	Product	Category	CategoryID	CategoryID
1	343	Road-650 Black, 52	Road Bikes	2	2
2	344	Mountain-100 Silver, 38	Mountain Bikes	1	1

INNER JOIN esegue due operazioni logiche: prima esegue un prodotto cartesiano. Dopo filtra le sole combinazioni di valori per cui risulta vero il predicato di join (che come il predicato della WHERE è sostanzialmente un filtro.. E filtra il prodotto cartesiano..!) In altre parole, il predicato di join è il criterio in base al quale verificare la corrispondenza dei valori. Il 'filtro' è introdotto dalla keyword **ON**

SQL: OUTER JOIN

L'**OUTER JOIN** consente di preservare tutti i record di una tabella e, combinare a questi i record della seconda tabella ogni qual volta si ha una corrispondenza (quando il predicato di join è vero).
Ogni qual volta non c'è una corrispondenza (predicato di join è falso) il risultato dell'**OUTER JOIN** join restituisce NULL!

Da un punto di vista logico l'operatore di JOIN esegue:

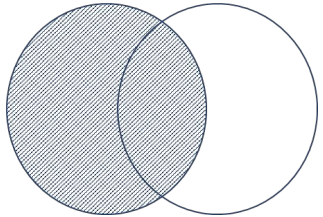
1) **Prodotto cartesiano – CROSS JOIN** (vengono restituite tutte le combinazioni possibili)
2) **Applica un filtro al prodotto cartesiano – INNER JOIN** (vengono restituite solo le combinazioni per cui è vero il predicato di join)
3) **'Aggiunge' righe esterne all'intersezione – OUTER JOIN**: vengono preservate tutte le righe della tabella di sinistra anche se non hanno una corrispondenza a destra (**LEFT OUTER JOIN)**; vengono preservate tutte le righe della tabella di destra anche se non hanno una corrispondenza a sinistra (**RIGHT OUTER JOIN)**

SQL: LEFT OUTER JOIN

LEFT OUTER JOIN consente di preservare **tutti** i record della tabella di sinistra e associare a questi i record della tabella sinistra per i quali esistente una corrispondenza (il predicato di join è vero!).
Supponiamo di dover restituire l'elenco di tutti i prodotti aziendali e la loro categoria (a prescindere dal fatto che l'abbiano o meno.. In altre parole, occorre restituire tutti i prodotti e la loro categoria anche se non ce l'hanno..!)

ProductID	Product	CategoryID
1	Adjustable Race	NULL
343	Road-650 Black, 52	2
344	Mountain-100 Silver, 38	1

CategoryID	Category
1	Mountain Bikes
2	Road Bikes
3	Touring Bikes



```
SELECT
P.ProductID
, P.Product
, C.Category
, C.CategoryID
, P.CategoryID
FROM Product AS P LEFT OUTER JOIN
Category AS C
ON P.CategoryID = C.CategoryID
```

ProductID	Product	CategoryID	CategoryID	Category
1	Adjustable Race	NULL	NULL	NULL
343	Road-650 Black, 52	2	2	Road Bikes
344	Mountain-100 Silver, 38	1	1	Mountain Bikes

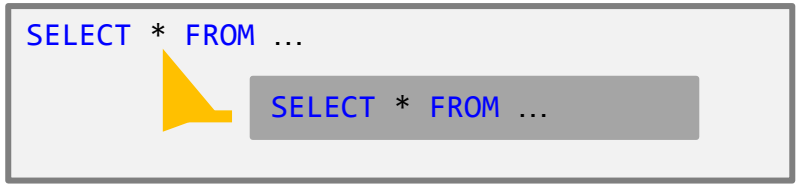
Il concetto di RIGHT OUTER JOIN è speculare!

SQL: SUBQUERIES

Le subquery sono query innestate cioè query all'interno di altre query.

Il risultato della query innestata (inner query) e utilizzato dalla query esterna (outer query).

In altre parole, il risultato della query interna è utilizzato come input dalla query esterna!



SQL: SUBQUERY SCALARE INDIPENDENTE

Le subquery sono query innestate cioè query all'interno di altre query.
Il risultato della query innestata (inner query) e utilizzato dalla query esterna (outer query).
In altre parole, il risultato della query interna è utilizzato come input dalla query esterna!

Supponiamo di dover interrogare la tabella sottostante per i codici prodotto il cui prezzo di listino è superiore al prezzo di listino medio (media della colonna ListPrice)

ProductID	Product	ListPrice
343	Road-650 Black, 52	782,99
344	Mountain-100 Silver, 38	3399,99
345	Mountain-100 Silver, 42	3399,99

```
SELECT ProductID, Product, ListPrice
FROM Prodotti2
WHERE ListPrice > (SELECT
AVG(ListPrice)
                  FROM Prodotti2)
```

Questo è un esempio di **query innestata scalare indipendente**: la query innestata (inner query) restituisce un valore (singolo valore) che viene utilizzato come input dalla query esterna (outer query).

È indipendente perché puo' essere eseguita indipendentemente dalla query esterna!
Una query innestata scalare puo' essere innestata in qualsiasi punto dello statament select!

	ProductID	Product	ListPrice
1	344	Mountain-100 Silver, 38	3399,99
2	345	Mountain-100 Silver, 42	3399,99

SQL: SUBQUERY AUTONOMA SCALARE

Le subquery sono query innestate cioè query all'interno di altre query.
Il risultato della query innestata (inner query) e utilizzato dalla query esterna (outer query).
In altre parole, il risultato della query interna è utilizzato come input dalla query esterna!

Supponiamo di dover interrogare la tabella sottostante per i codici prodotto il cui prezzo di listino è superiore al prezzo di listino medio (media della colonna ListPrice)

ProductID	Product	ListPrice
343	Road-650 Black, 52	782,99
344	Mountain-100 Silver, 38	3399,99
345	Mountain-100 Silver, 42	3399,99

```
SELECT
ProductID
, Product
, ListPrice
, (SELECT AVG(ListPrice)
   FROM Prodotti2) AS AvgPrice
FROM Prodotti2
WHERE ListPrice > (SELECT AVG(ListPrice)
   FROM Prodotti2)
```

Questo è un esempio di **query innestata scalare indipendente**: la query innestata (inner query) restituisce un valore (singolo valore) che viene utilizzato come input dalla query esterna (outer query).

È indipendente perché puo' essere eseguita indipendentemente dalla query esterna!
Una query innestata scalare puo' essere innestata in qualsiasi punto dello statament select!

	ProductID	Product	ListPrice	AvgPrice
1	344	Mountain-100 Silver, 38	3399,99	2527,6566
2	345	Mountain-100 Silver, 42	3399,99	2527,6566

SQL: SUBQUERY AUTONOMA MULTIVALORE

Le subquery sono query innestate cioè query all'interno di altre query.
Il risultato della query innestata (inner query) e utilizzato dalla query esterna (outer query).
In altre parole, il risultato della query interna è utilizzato come input dalla query esterna!

Supponiamo di dover interrogare la tabella sottostante per i codici prodotto la cui categoria sia *Road Bikes* o *Mountain Bikes*

ProductID	Product	CategoryID
1	Adjustable Race	NULL
343	Road-650 Black, 52	2
344	Mountain-100 Silver, 38	1

CategoryID	Category
1	Mountain Bikes
2	Road Bikes
3	Touring Bikes

Le **sottoquery multivalore indipendenti** restituiscono un risultato in modo molto simile a una tabella a colonna singola (o ad una lista). Sono utili quando la query esterna necessità NON di un singolo valore ma quando deve elaborare più valori. È particolarmente adatta all'uso in una condizione di ricerca con l'operatore IN (la query esterna ricerca valori all'interno di una 'lista' o colonna di valori!!

```
SELECT *
FROM Product
WHERE CategoryID IN
                    (SELECT CategoryID
                     FROM Category
                     WHERE Category LIKE 'Road Bikes'
                     OR Category LIKE 'Mountain Bikes')
```

	ProductID	Product	CategoryID
1	343	Road-650 Black, 52	2
2	344	Mountain-100 Silver, 38	1

SQL

LET'S TAKE A LOOK!

EPICODE

GRAZIE

Epicode