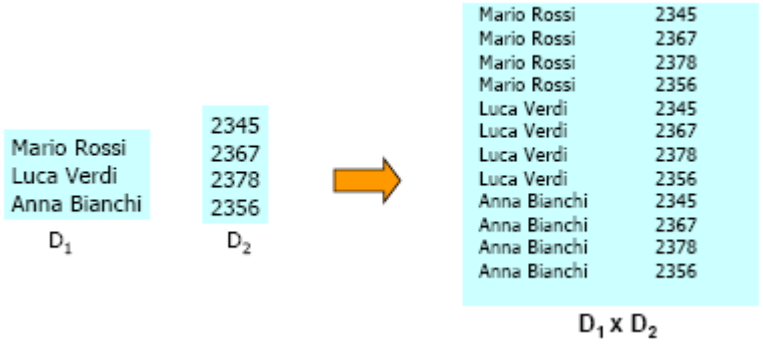
Operazioni dell'algebra relazionale

Le istruzioni SQL vengono scomposte dal DBMS in una serie di operazioni relazionali.

Le operazioni **primitive** dell'algebra relazionale sono:

- **Prodotto** (o prodotto cartesiano) di relazioni ottenuto "*concatenando*" ogni tupla di una relazione con ogni tupla dell'altra. Si intende per ***concatenazione*** tra due tuple $(x_1, \ldots x_n)$ e $(y_1, \ldots y_n)$ la tupla $(x_1, \ldots x_n, y_1, \ldots y_n)$

  Il risultato è una relazione con tutte le possibili associazioni tra gli elementi degli insiemi



In linguaggio **SQL** si usa l'operatore **virgola** nella clausola from

- **Proiezione** di una relazione per selezionare solo determinate *colonne* da una tabella
- **Restrizione** di una relazione o selezione per estrarre delle tuple da una relazione in base ad un criterio
- **Ridenominazione** di un attributo per evitare conflitti
- **Unione** di relazioni ad esempio per disporre di un catalogo completo si uniscono insiemisticamente le relazioni Articoli_Vecchi ed Articoli_Nuovi
- **Differenza** di relazioni ad esempio per isolare le tuple relative agli Articoli_Nuovi si sottraggono insiemisticamente le relazioni Articoli ed Articoli_Vecchi

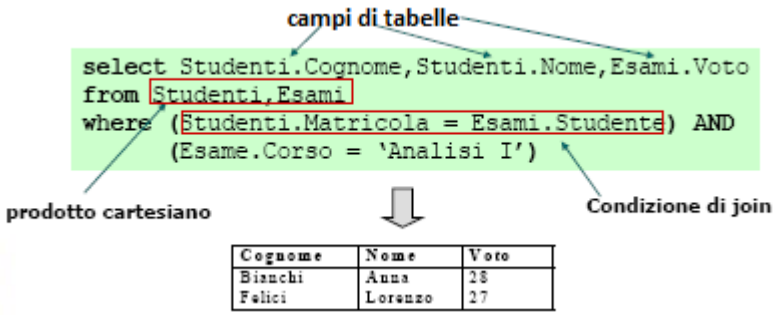
Giunzione interna

Tra le operazioni **derivate** (cioè ottenibili da una combinazione di quelle primitive elencate), rivestono maggior utilità pratica quelle di **giunzione**[1] (***join***) che consentono di costruire una relazione, partendo da due relazioni, sulla base di qualche criterio. L'uso di giunzioni permette di selezionare dati prelevandoli fra più tabelle, evidentemente correlate tra loro.

L'operazione di **giunzione interna** (inner join) fissa come criterio di selezione delle tuple una **condizione qualsiasi** tra due campi relativi alle due diverse relazioni: si cercano righe corrispondenti nelle due tabelle, basandosi sul valore di determinate colonne. Da questo punto di vista, il prodotto di relazioni può essere visto come un caso particolare di giunzione interna (cross) in cui si specifica la condizione *true*.

Caso particolare di giunzione interna è l'operazione di **θ-join** dove **θ** è un **operatore di confronto** (<, >, =, ecc) applicato sui due campi.

Caso particolare di θ-join è l'**equigiunzione** (***equijoin***) che corrisponde a mantenere solo quelle righe in cui risultano **uguali** i due campi (A della relazione *R* e B della relazione *S*). Il risultato conterrà per forza due colonne (A e B) uguali. Per evitare questa situazione si può ricorrere all'operazione di giunzione naturale.



[1] Nb: l'operazione di giunzione è quella più critica dal punto di vista dell'efficienza (il tempo di esecuzione potrebbe risultare inaccettabile in molte applicazioni)

L'operazione di **giunzione naturale** (***natural join**:* basata su tutte le *colonne* che nelle due tabelle hanno lo *stesso nome*) viene illustrata nell'ipotesi che vi sia un solo campo in comune (è facilmente estendibile a più campi comuni): si rinomina il campo comune in una delle due relazioni, si fa l'equijoin rispetto ai due campi e si elimina una delle colonne che risultano uguali.

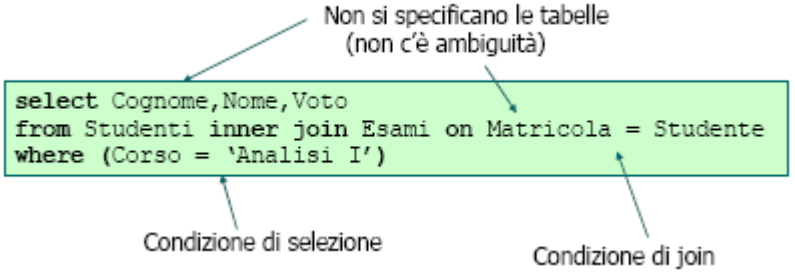
Semi-giunzione (***semi-join***) è come una giunzione naturale ma vengono proiettati solo i campi della relazione di sinistra.

Self-join: si usa quando nell'interrogazione è necessario riferirsi a diverse istanze di una stessa tabella e dopo averla ridenominata si realizza in pratica una giunzione tra la tabella e sé stessa

SQL-2 ha introdotto una sintassi alternativa per i **join**, rappresentadoli esplicitamente nella clausola from:

SELECT *AttrEspr* [[AS] *Alias*] {, *AttrEspr* [[AS] *Alias*] }
FROM *Tabella* [[AS] *Alias*]
{ [*TipoJoin*[2]] **JOIN** *Tabella* [[AS] *Alias*] **ON** *Condizioni* }
[WHERE *AltreCondizioni*]

Si ricordi che in Base l'inner join è realizzato come select con from che si riferisce a due tabelle e clausola where che confronta colonne delle diverse tabelle (prodotto con eliminazione delle righe che non soddisfano la condizione e proiezione ottenuta sui campi selezionati).

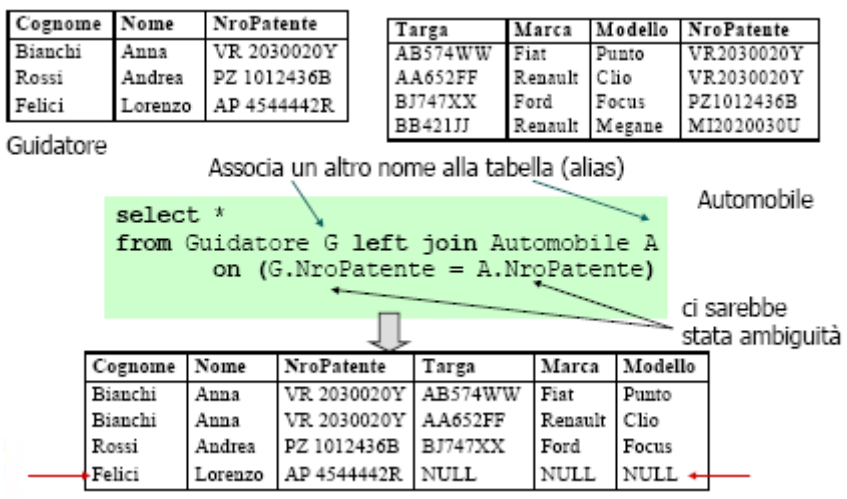


Giunzioni esterne (***outer-join***)

Operazioni non derivabili dalle primitive, che appaiono come estensioni della giunzione naturale. Introdotte per unire tabelle preservando, nel risultato, tutte le informazioni presenti su una o entrambe le tabelle di partenza. La differenza, rispetto ad una giunzione interna, è la possibilità di estrarre anche le righe di una tabella che **non** hanno corrispondenti nell'altra. I campi che risulterebbero indefiniti vengono riempiti con valore nullo. Normalmente si considerano tre tipi di giunzioni esterne:

- ***left outer join*** (si preservano le tuple left cioè otterremo le righe senza corrispondente che si trovano nella tabella di sinistra, quella dichiarata per prima nella query),
- ***right outer join*** (si preservano le tuple right),
- ***full outer join*** (si preservano le tuple di entrambe).

La variante OUTER può essere utilizzata sia per join naturale che per theta-join



Union join: esiste un ulteriore operatore di join, UNION JOIN, tale che R UNION JOIN S contiene ogni colonna e ogni riga di R e di S, completate con NULL in tutti i campi non presenti nella tupla originaria

[2] *TipoJoin* può essere inner, right [outer], left [outer] oppure full [outer], consentendo la rappresentazione dei join esterni. La parola chiave natural può precedere *TipoJoin* (però è implementato di rado). *In MySQL sono supportate solo le JOIN interne e la JOIN a sinistra/destra o esterna*

Join in ACCESS

Ad esempio (*impiego.mdb*): Tabella Impiegato

cf	nome	pp
0	Rossi	1000
1	Verro	1200
2	Bianchi	1400
3	Verdi	1200
4	Gialli	1000

Tabella Lavora

rid	cf	data
0	0	04/11/07
0	3	04/11/06
0	4	03/10/05
1	1	12/12/07
2	2	02/11/07

SELECT d1.nome, d1.pp, d2.rid **AS** idReparto
FROM Impiegato **AS** d1, Lavora **as** d2
WHERE d1.cf = d2.cf

effetto:

	nome	pp	idReparto
▶	Rossi	1000	0
	Verro	1200	1
	Bianchi	1400	2
	Verdi	1200	0
	Gialli	1000	0

analogo a

SELECT d1.nome, d1.pp, d2.rid **AS** idReparto
FROM Impiegato d1 **INNER JOIN** Lavora d2
ON d1.cf = d2.cf

La sintassi del join è la seguente:

```
SELECT d1.nome, d1.cognome, d2.nome AS nomeCapo, d2.cognome AS cognomeCapo

FROM dipendente d1 JOIN dipendente d2 ON d1.capo = d2.cf
```

La tabella risultato dell'operazione di join è detta **tabella congiunta** (*joined table*). Le tabelle argomento dell'operazione di join possono essere loro stesse tabelle congiunte.

Ad esempio, il seguente join recupera i dipendenti con stipendio inferiore a 1000 e i capi dei loro capi:

```
SELECT d1.nome, d1.cognome, d3.nome AS nomeSuperCapo, d3.cognome AS cognomeSuperCapo

FROM (dipendente d1 JOIN dipendente d2 ON d1.capo = d2.cf)  JOIN dipendente d3
ON d2.capo = d3.cf

WHERE d1.stipendio < 1000
```

Join fra più tabelle

Abbiamo visto esempi di join fra due tabelle, ma è possibile effettuarne anche fra più di due. In questo caso l'operazione sarà logicamente suddivisa in più join, ciascuna delle quali viene effettuata fra due tabelle; il risultato di ogni join diventa una delle due tabelle coinvolte nella join successiva. L'ordine con cui vengono effettuate le diverse join dipende dall'ordine in cui sono elencate le tabelle e dall'uso di eventuali parentesi:

FROM *t1* **JOIN** *t2* **ON** t1.col1 = t2.col2 **LEFT JOIN** *t3* **ON** t2.col3 = t3.col3

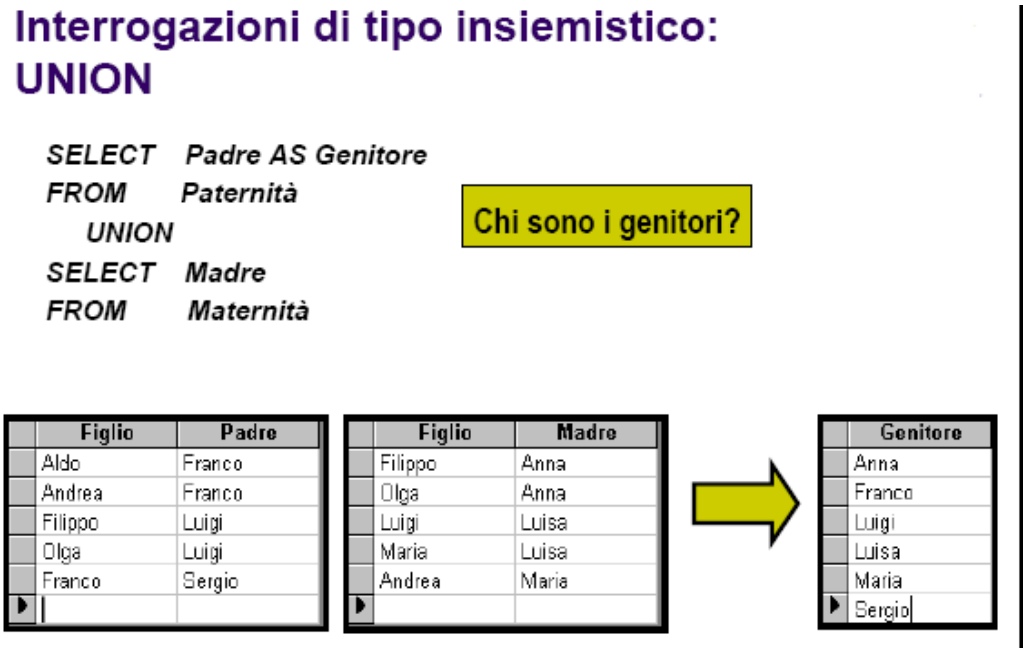
In questo caso viene effettuata prima la join fra t1 e t2; di seguito, il risultato di questa join viene utilizzato per la left join con t3.

FROM *t1* **JOIN (***t2* **LEFT JOIN** *t3* **ON** t2.col3 = t3.col3**) ON** t1.col1 = t2.col2

La presenza delle parentesi fa sì che venga effettuata prima la left join fra t2 e t3, e di seguito il risultato venga utilizzato per la inner join con t1.

Se le relazioni su cui si opera hanno una s**truttura tabellare omogenea**, cioè colonne con lo stesso numero di campi, dello stesso tipo e nello stesso ordine, si possono applicare le usali **operazioni sugli insiemi**:

- **Unione** che genera a partire da due tabelle omogenee una nuova tabella che contiene le righe di entrambe con riduzione a una di quelle ripetute
- **Intersezione** che genera a partire da due tabelle omogenee una nuova tabella che contiene solo le righe comuni
- **Differenza** che genera a partire da due tabelle omogenee una nuova tabella che contiene solo le righe della prima non contenute nella seconda



Per implementare **Intersezione** o **Differenza** in ACCESS si ricorre a *interrogazioni nidificate* ed uso rispettivamente di predicati **IN** e **NOT IN**.