ALTRI OPERATORI DERIVATI

Le t-uple **DANGLING** sono le t-uple pendenti che non partecipano alla join, quindi quelle "scartate".

Per esempio:

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
production	Mori
purchasing	Brown

$r_1 \bowtie r_2$

Employee	Department	Head
Black	production	Mori
White	production	Mori

Le tuple dangling, in questo caso, sono:

- Smith | sales dalla tabella di sinistra;
- purchasing | Brown dalla tabella di destra.

Casi estremi

Potrebbe anche succedere che **nessuna n-upla trovi il corrispettivo** nella seconda tabella. Pertanto il **risultato** della JOIN sarà la **tabella vuota.**

Esempio:

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
marketing	Mori
purchasing	Brown

$r_1 \bowtie r_2$

Employee	Department	Head

Ogni n-upla della prima tabella r1 si combina con ogni n-upla di r2 quindi il risultato della join sarà il **prodotto cartesiano**.

> Quindi si possono simulare i 2 casi estremi (tabella vuota o prodotto cartesiano)

> La **cardinalità** del risultato è il prodotto delle cardinalità

r_1

Employee	Project
Smith	A
Black	A
White	A

r_2

Project	Head
A	Mori
A	Brown

$r_1 \bowtie r_2$

Employee	Project	Head
Smith	A	Mori
Black	A	Brown
White	A	Mori
Smith	A	Brown
Black	A	Mori
White	A	Brown

Outer JOIN

L'outer join, cioè la *giunzione esterna*, è una variante della join che serve per **mantenere le tuple dangling che non partecipano alla join**. Ci sono 3 varianti di quest'operazione:

- **Left**, riempie di NULL gli attributi dell'operando a destra e ne fa la JOIN;
- **Right**, riempie di NULL gli attributi dell'operando a sinistra e ne fa la JOIN;
- **Full**, vengono riempite di NULL gli attributi mancanti a destra e a sinistra e si effettua la JOIN.

> Ogni join può essere di tipo left, right oppure full join.

La giunzione esterna è la **giunzione naturale estesa** con tutte le n-uple che non appartengono alla giunzione naturale, completate con valori NULL per gli attributi mancanti.

Siano R ed S definite sugli insiemi di attributi XY e YZ rispettivamente.

$$R \overleftrightarrow{\bowtie} S = (R \bowtie S) \cup ((R - \pi_{XY}(R \bowtie S)) \times \{Z = NULL\}) \cup (\{X = NULL\} \times (S - \pi_{YZ}(R \bowtie S)))$$

- la **freccia** sopra il simbolo JOIN mi dice in **quale direzione riempio le tabelle**

join fra R e S U (tuple tabelle sinistre che non sono prese dalla join) X =null U a sinistra metto null e a destra le tuple che mancavano

- right join è R join S e la seconda parte della formula, cioè x null e s- proiezione su yz di r ed s
- analogamente per la left.

- Definiamo Giunzione Esterna Sinistra:

$$R \overleftarrow{\bowtie} S = (R \bowtie S) \cup ((R - \pi_{XY}(R \bowtie S)) \times \{Z = NULL\})$$

- Definiamo Giunzione Esterna Destra:

$$R \overrightarrow{\bowtie} S = (R \bowtie S) \cup (\{X = NULL\} \times (S - \pi_{YZ}(R \bowtie S)))$$

Esempio

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
production	Mori
purchasing	Brown

$r_1 \bowtie_{LEFT} r_2$

Employee	Department	Head
Smith	Sales	NULL
Black	production	Mori
White	production	Mori

$r_1 \bowtie_{RIGHT} r_2$

Employee	Department	Head
Black	production	Mori
White	production	Mori
NULL	purchasing	Brown

$r_1 \bowtie_{FULL} r_2$

Employee	Department	Head
Smith	Sales	NULL
Black	production	Mori
White	production	Mori
NULL	purchasing	Brown

- Il JOIN e'
  - Commutativo: $R \bowtie S = S \bowtie R$
  - Associativo: $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$
- Quindi possiamo avere sequenze di JOIN senza rischio di ambiguita:

$$R \bowtie S \bowtie T \ldots$$

- In caso di mancate parentesi non cambia nulla visto che la proprietà è associativa
- Se ho **left** o **right** join essi **NON** sono commutativi.

Esempio operazioni multiple

r_1

Employee	Department
Smith	sales
Black	production
Brown	marketing
White	production

r_2

Department	Division
production	A
marketing	B
purchasing	B

r_2

Division	Head
A	Mori
B	Brown

$r_1 \bowtie r_2 \bowtie r_3$

Employee	Department	Division	Head
Black	production	A	Mori
Brown	marketing	B	Brown
White	production	A	Mori

Join è come un doppio ciclo for quindi conviene fare join (in caso di join multiple) fra tabelle con minori record per ottimizzare.

> Per **convenzione join è definita anche se non ci sono attributi in comune e il risultato corrisponde al prodotto cartesiano**

Esempio:

Employees

Employee	Project
Smith	A
Black	A
Black	B

Projects

Code	Name
A	Venus
B	Mars

Employes $\bowtie$ Projects

Employee	Project	Code	Name
Smith	A	A	Venus
Black	A	A	Venus
Black	B	A	Venus
Smith	A	B	Mars
Black	A	B	Mars
Black	B	B	Mars

- **Per convenzione se le 2 relazioni hanno gli stessi attributi si ottiene con la join l'intersezione fra gli attributi**
- semi join: join dove vado a prendere già attributi solo della tabella di sinistra.

  - Siano R con attributi XY ed S con attributi YZ
  - $R \ltimes S$ è una relazione di attributi XY costituita da tutte le n-uple di R che partecipano a $R \bowtie S$.
  - La semi-giunzione e' derivata perché

$$R \ltimes S = \pi_{XY}(R \bowtie S)$$

Nome	Matricola	Indirizzo	Telefono
Mario Rossi	123456	Via Etnea 1	222222
Ugo Bianchi	234567	Via Roma 2	333333
Teo Verdi	345678	Via Torino 3	444444

Corso	Matricola	Voto
Architettura	123456	30
Programmazione	234567	18
Architetture	234567	27

Studenti ⋉ Esami

Nome	Matricola	Indirizzo	Telefono
Mario Rossi	123456	Via Etnea 1	222222
Ugo Bianchi	234567	Via Roma 2	333333

> Si proiettano solo gli attributi della tabella di sinistra nel risultato

Unione esterna

- Siano R ed S due relazioni definite sugli insiemi di attributi XY e YZ allora
- L'unione esterna

$$R \overleftrightarrow{\cup} S = R \times \{Z = NULL\} \cup \{X = NULL\} \times S$$

- si ottiene estendendo le due tabelle con le colonne dell'altra con valori nulli e si fa l'unione.

Prendo la tabella di destra (e sinistra) e negli attributi che mancano metto *NULL* e faccio unione di entrambe.

A	B	C	D
X1	Y	Z	X
X2	Y	Z	X
X3	Y	W	X
X4	Y	W	X

R

B	C	D	E
Y	Z	X	Y1
Y	Z	X	M1
Y	W	X	Y2
Y	W	X	M2

S

$R \overleftrightarrow{\cup} S$

A	B	C	D	E
X1	Y	Z	X	NULL
X2	Y	Z	X	NULL
X3	Y	W	X	NULL
X4	Y	W	X	NULL
NULL	Y	Z	X	Y1
NULL	Y	Z	X	M1
NULL	Y	W	X	Y2
NULL	Y	W	X	M2

> Per fare l'unione fra attributi non in comune posso usare l'unione esterna per mantenere gli attributi non in comune

---

Si presenta una **problematica** con valori nulli con operazioni di selezione

Impiegati

Matricola	Cognome	Filiale	Età
7309	Rossi	Roma	32
5998	Neri	Milano	45
9553	Bruni	Milano	NULL

$$\sigma_{\text{Età} > 40}\,(\text{Impiegati})$$

- la condizione atomica è vera solo per valori non nulli

- Come si prende in considerazione la riga con null?
- Ci 2 nuovi operatori IS NULL e IS NOT NULL (presenti in SQL)

$$\sigma_{\text{Età} > 40}\,(\text{Impiegati})$$

- La condizione atomica è vera solo per valori non nulli
- Per riferirsi ai valori nulli esistono forme apposite di condizioni:

  IS NULL

  IS NOT NULL
- si potrebbe usare (ma non serve) una "logica a tre valori" (vero, falso, sconosciuto)

Con questo operatore si usa la logica booleana a 3 valori e non a 2 come accadeva per la booleana classica a 2 valori.

Cioè:

p	q	p and q	p or q	not p
T	T	T	T	F
T	F	F	T	F
T	U	U	T	F
F	F	F	F	T
F	U	F	U	T
U	U	U	U	U

- Quindi:

$$\sigma_{Età>30}(Persone) \cup \sigma_{Età\leq 30}(Persone) \cup \sigma_{Età\ IS\ NULL}(Persone)$$

$$=$$

$$\sigma_{Età>30\ \vee\ Età\leq 30\ \vee Età\ IS\ NULL}(Persone)$$

$$=$$

$$Persone$$

ESEMPIO:

Vendite

VENDITORI	CITTÀ
v1	CT
v2	RM
v1	PA
v3	ME
v2	PA
v3	RM
v2	CT
v3	ME

Città

CITTÀ
CT
RM
PA
ME

> Come trovo i venditori che hanno venduto in tutte le città?

$\pi_{venditore}((\pi_{venditore}(Vendite)$X Città)\ Vendite) = R1 (venditori che non hanno venduto in qualche città)

$\pi_{venditore}Vendite$\R1 i venditori che hanno venduto in tutte le città

- Calcolo prima tutte le possibili coppie
- Tolgo tutte quelle coppie che sono già presenti in Vendite

- Restano le v1 che non ha venduto a RM e ME, v3 che non ha venduto a CT e PA
- fra tutti i venditori tolgo questi, ottengo **chi ha venduto in tutte le città.**

Questo è un operatore derivato e si chiama **QUOZIENTE**.
Vendite / Città esegue esattamente l'operazione vista sopra.

- **Divisione:** Siano XY gli attributi di R ed Y quelli di S, allora

$$R \div S = \{w | \{w\} \times S \subseteq R\}$$

La divisione serve a rispondere a query del tipo:

> trova **TUTTE** le n-uple di R associate a **TUTTE** le n-uple di S.

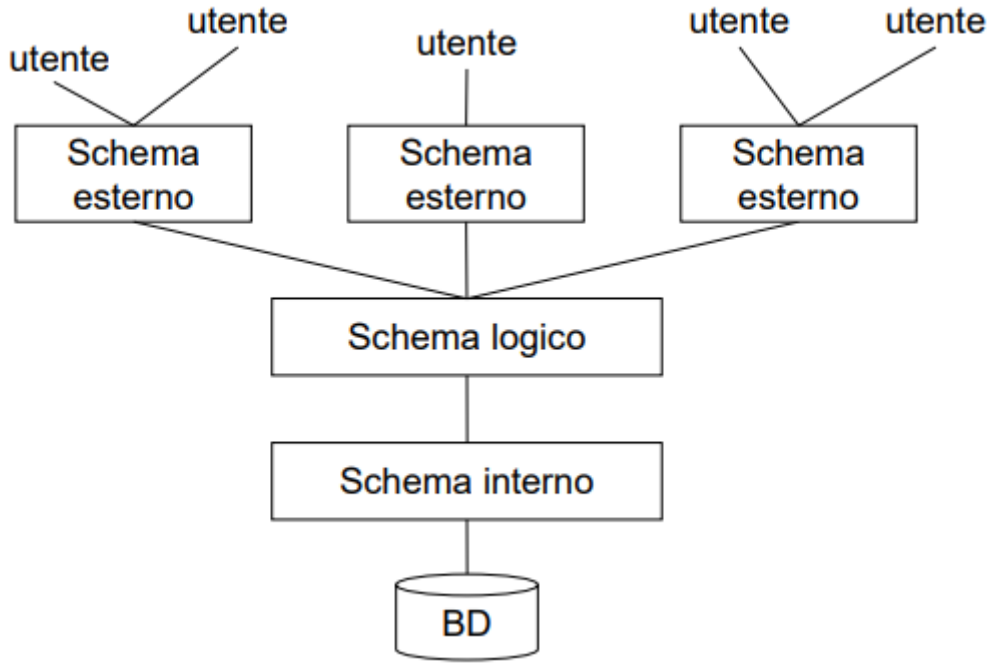
Viste

- Le **relazioni di base** hanno un **contenuto autonomo** di partenza del database.
- Le **relazioni derivate** il contenuto è funzione di altro contenuto, definita sulla base di query

$\pi_{venditore}((\pi_{venditore}(Vendite)$X Città)\ Vendite) = R1 è una relazione derivata, cioè una VISTA

> la vista è una tabella generata da una query

Schema esterno, fisico, logico ecc
Viste= schema esterno



Il procedimento, quindi, è questo:

- query->nome->vista

Esistono due tipi di relazioni derivate:

- **viste materializzate**: si effettua anche copia del dato(tabella) secondo quella specificata. Sono **immediatamente disponibili** ma c'è una copia **ridondante dei dati**:
  - inserisco 1 nuovo record-> il DB prende tutte le viste materializzate e le ricalcola dall'inizio
- **relazioni virtuali** (o viste): un rimpiazzo per quella query quindi non c'è una copia ma proprio la sostituisce, come un segnaposto per la query. In questo caso si deve **eseguire ogni volta la query MA** in pratica non si rallenta il DB per operazioni di inserimento/cancellazione: se aggiungo 1 nuovo record il DB non deve ricalcolare nulla.

> Tutti i DB implementano le viste, quindi vi è un segnaposto per la query

- Sono eseguite sostituendo alla vista la sua definizione:

$$\sigma_{Capo='Leoni'}\ (Supervisione)$$

viene eseguita come

$$\sigma_{Capo='Leoni'}(\ \pi_{Impiegato,\ Capo}\ (Afferenza \bowtie Direzione))$$

Le viste le definiamo dando un nome ad una query e si usano come se fossero tabelle a tutti gli effetti con un nome specifico, tipo R1.

- Schema esterno: ogni utente vede solo
  - ciò che gli interessa e nel modo in cui gli interessa, senza essere distratto dal resto
  - ciò che e' autorizzato a vedere (autorizzazioni)
- **Strumento di programmazione**:
  - si può semplificare la scrittura di interrogazioni: espressioni complesse e sottoespressioni ripetute
- Utilizzo di programmi esistenti su schemi ristrutturati

Invece:

- L'utilizzo di viste non influisce sull'efficienza delle interrogazioni

Viste come strumento di programmazione

- Trovare gli impiegati che hanno lo stesso capo di Rossi

- Senza vista:

$$\pi_{Impiegato}\ ((Afferenza \bowtie Direzione) \bowtie \delta_{ImpR,RepR \leftarrow Imp,Reparto}\ (\sigma_{Impiegato='Rossi'}\ (Afferenza \bowtie Direzione)))$$

- Con la vista:

$$\pi_{Impiegato}\ (Supervisione \bowtie \delta_{ImpR,RepR \leftarrow Imp,Reparto}\ (\sigma_{Impiegato='Rossi'}\ (Supervisione)))$$