Lezione 10

Operatori relazionali e SQL

Operazioni del modello relazionale

- Le operazioni del modello relazionale sono di due tipi:
  - di modifica
    - *updates*
  - di interrogazione per il recupero delle informazioni
    - *query* e *retrieval*
- Modalità di presentazione
  - definizione formale sul modello relazionale
  - realizzazione in SQL

Operazioni insiemistiche

- Siano r_1 e r_2 due relazioni definite sullo **stesso** insieme di attributi $X=\{A_1,A_2,\ldots,A_n\}$
- è possibile definire le seguenti operazioni:
  - **Unione di r_1 e r_2**: una nuova relazione r ottenuta considerando le tuple di r_1 e r_2

$$r = r_1 \cup r_2 = \{t : t \in r_1 \vee t \in r_2\}$$

  - **Intersezione di r_1 e r_2**: una nuova relazione r ottenuta considerando le tuple di r_1 che sono anche tuple di r_2 e viceversa

$$r = r_1 \cap r_2 = \{t : t \in r_1 \wedge t \in r_2\}$$

  - **Differenza di r_1 e r_2**: una nuova relazione r ottenuta considerando le tuple di r_1 escluse quelle che appartengono anche ad r_2

$$r = r_1 - r_2 = \{t : t \in r_1 \wedge t \notin r_2\}$$

Esempio unione

Matricola	Nome	Cognome	Indirizzo
150	Alex	Del Piero	via dei Palloni, 30
151	Martina	Stellina	via del Cielo, 40
142	Giovanni	SenzaTerra	via delle Crociate, 30

r1 - Studenti di Basi di Dati

Matricola	Nome	Cognome	Indirizzo
142	Giovanni	SenzaTerra	via delle Crociate, 30
149	Marco	Rossi	via dei colori, 40

r2 - Studenti di Analisi I

$r_1 \cup r_2$

Matricola	Nome	Cognome	Indirizzo
150	Alex	Del Piero	via dei Palloni, 30
151	Martina	Stellina	via del Cielo, 40
142	Giovanni	SenzaTerra	via delle Crociate, 30
149	Marco	Rossi	via dei colori, 40

Studenti che frequentano o il corso di Basi di Dati o il corso di Analisi I

Esempio intersezione

Matricola	Nome	Cognome	Indirizzo
150	Alex	Del Piero	via dei Palloni, 30
151	Martina	Stellina	via del Cielo, 40
142	Giovanni	SenzaTerra	via delle Crociate, 30

r1 - Studenti di Basi di Dati

Matricola	Nome	Cognome	Indirizzo
142	Giovanni	SenzaTerra	via delle Crociate, 30
149	Marco	Rossi	via dei colori, 40

r2 - Studenti di Analisi I

$r_1 \cap r_2$

Matricola	Nome	Cognome	Indirizzo
142	Giovanni	SenzaTerra	via delle Crociate, 30

Studenti che frequentano sia il corso di Basi di Dati sia il corso di Analisi I

Esempio differenza

Matricola	Nome	Cognome	Indirizzo
150	Alex	Del Piero	via dei Palloni, 30
151	Martina	Stellina	via del Cielo, 40
142	Giovanni	SenzaTerra	via delle Crociate, 30

r1 - Studenti di Basi di Dati

Matricola	Nome	Cognome	Indirizzo
142	Giovanni	SenzaTerra	via delle Crociate, 30
149	Marco	Rossi	via dei colori, 40

r2 - Studenti di Analisi I

$r_1 - r_2$

Matricola	Nome	Cognome	Indirizzo
150	Alex	Del Piero	via dei Palloni, 30
151	Martina	Stellina	via del Cielo, 40

Studenti che frequentano il corso di Basi di Dati e non il corso di Analisi I

Operazioni di modifica della base di dati

- Premessa:
  - Il modello proposto è valido nel caso di operazioni che coinvolgono una sola tupla di una data relazione
  - Dopo l'introduzione delle query SQL tale modello sarà esteso ad operazioni che coinvolgono più tuple

- Operazioni considerate:
  - **insert** – inserimento di una nuova tupla all'interno di una relazione
  - **delete** – cancellazione di una tupla da una relazione
  - **update** – modifica dei valori di una tupla all'interno di una relazione

Inserimento

- Un'operazione di **inserimento** permette di inserire una nuova tupla all'interno di una data relazione *REL* definita su un insieme di attributi $X=\{A_1,A_2,\ldots,A_n\}$ con $v_i \in dom(A_i)$ *per i che va da 1 a n*

$$\text{INSERT} ::= \langle v_1, v_2, \ldots v_n \rangle \bigcup \text{REL} \mapsto \text{REL}$$

- L'inserimento può causare la violazione di:
  - Vincoli di dominio
  - Vincoli di chiave primaria
  - Vincoli di integrità referenziale

- …a valle della violazione di un vincolo di norma un DBMS rifiuta l'operazione

Inserimento in SQL

INSERT INTO nome_tab[(lista attributi)]
VALUES (lista valori)

- Inserimento di un nuovo studente del corso di Basi di Dati con tutte le informazioni che lo riguardano

  INSERT INTO STUDENTI_BASIDATI **VALUES** (155, 'Carla', 'Fracci', 'via del Campo, 21')

- Inserimento di un nuovo studente del corso di Basi di Dati con le sole informazioni relative a matricola e nome

  INSERT INTO STUDENTI_BASIDATI(Matricola, Nome) **VALUES** (160, 'Carla')

Considerazioni

- La lista_attributi è opzionale quando si specificano i valori di tutti gli attributi
- la corrispondenza tra lista attributi e lista valori è per posizione
- se alcuni valori non vengono specificati il DBMS assegna il valore di default
- NULL assegna valore nullo

Esempi

Dipartimento(Codice:char(4),Nome:varchar(20))

- Insert into Dipartimento
  values(‘aaa’,’Amministrazione’)
- Insert into Dipartimento(Codice,Nome)
  values(‘aaa’,’Amministrazione’)
- Insert into Diparttimento(Codice) values (‘cccc’)
- Insert into Diparttimento(Codice,Nome)
  values (‘dddd’,NULL)

Cancellazione

- Un'operazione di **cancellazione** elimina una tupla da una data relazione *REL* definita su un insieme di attributi $X=\{A_1,A_2,\ldots,A_n\}$ con $v_i \in dom(A_i)$

$$\text{DELETE} ::= \text{REL} - \langle v_1, v_2 \ldots v_n \rangle \mapsto \text{REL}$$

- La cancellazione può causare la violazione di un vincolo di integrità referenziale, nel qual caso il DBMS si comporta:
  - *Rifiuto della cancellazione* (*no action* o *restrict*)
    - la cancellazione non viene eseguita
  - *Propagazione in cascata* (*cascade*)
    - vengono cancellate in cascata tutte le tuple delle tabelle referenti che referenziano la tupla cancellata nella tabella riferita
  - *Modifica di un valore negli attributi referenti* (*set*)
    - ogni valore nella relazione referente è posto a NULL (se non esiste un vincolo di NOT NULL), oppure ad un valore fissato

Cancellazione in SQL

DELETE FROM nome_tab WHERE condizione

- Elimina dalla tabella nome_tab tutte le tuple che rendono vera la condizione

```
DELETE FROM STUDENTI_BASIDATI WHERE Matricola=155
```

- Cancella lo studente del corso di Basi di Dati avente matricola pari a 155

Esempi

Dipartimento(Codice:char(4),Nome:varchar(20))

- Delete from Dipartimento where Nome is Null

- Se la condizione where manca allora la tabella viene svuotata.
  - delete from Dipartimento

- Da non confondere con:
  - drop table Dipartimento

  che elimina la tabella

Modifica

- Un'operazione di **modifica** cambia i valori su uno o più attributi di una tupla di una data relazione *REL* definita su un insieme di attributi $X=\{A_1,A_2,\ldots,A_n\}$ con $v_i \in dom(A_i)$

$$\text{UPDATE} ::= \text{REL} - \langle v_1, v_2 \ldots v_n \rangle \mapsto \text{REL}$$
$$\langle v_1, v_2 \ldots v_n \rangle \bigcup \text{REL} \mapsto \text{REL}$$

- In presensa di modifica il DBMS deve:
  - verificare il rispetto dei vincoli di dominio
  - se la modifica coinvolge una chiave primaria, si hanno effetti simili alle operazioni combinate di DELETE e INSERT
  - se la modifica coinvolge una chiave esterna il DBMS deve verificare che il nuovo valore nella tabella referente sia presente nella tabella riferita, oppure sia NULL

Modifica in SQL

UPDATE nome_tab SET attributo=espressione[,attributo=espressione]
[WHERE condizione]

- Se la clausola WHERE è omessa la modifica si estende a tutte le tuple della tabella

```
UPDATE STUDENTI_BASIDATI SET Nome='Carletta', Cognome='Fraccini'
WHERE Matricola=155
```

Modifica del nome e del cognome dello studente del corso di Basi di Dati avente matricola pari a 155

Esempi

Dipartimento(Codice:char(4),Nome:varchar(20))

- Update Dipartimento Set Nome = ‘Amministrazione’
- Update Dipartimento Set Nome = ‘Amministrazione’ where Codice = ‘aaaa’
- Update Dipartimento Set Nome = ‘Sconosciuto’ where Nome is Null
- UPDATE prodotto SET prezzo = prezzo * 1.10

Operatori relazionali

- Operazioni classiche del modello relazionale:
  - **Proiezione**
  - **Selezione**
  - **Ridenominazione**
  - **Prodotto cartesiano**
  - **Join**

- Approccio proposto:
  - ad un descrizione formale di tipo procedurale delle operazioni segue la descrizione della sintassi SQL

Proiezione

- Sia dato uno schema di relazione *R(X)*; sia *Y* un sottoinsieme di attributi di *X* e sia *r* una istanza di R. Si definisce **proiezione** della relazione *r* rispetto a *Y* l'insieme dei soli valori di *Y* di tutte le tuple t della relazione

$$\pi_{\mathbf{Y}}(r) = \{t[Y], t \in r\}$$

Esempio

Matricola	*Nome*	*Cognome*	*Indirizzo*
150	Alex	Del Piero	via dei Palloni, 30
151	Martina	Stellina	via del Cielo, 40
142	Giovanni	SenzaTerra	via delle Crociate, 30

r_1 - Studenti di Basi di Dati

$$\pi_{matricola,nome,cognome}(studenti)$$

Matricola	*Nome*	*Cognome*
150	Alex	Del Piero
151	Martina	Stellina
142	Giovanni	SenzaTerra

Osservazioni

- Si noti che se dopo la proiezione si hanno due tuple uguali (e potrebbe capitare se la proiezione non contiene attributi chiave), allora le tuple ripetute vengono fuse in un'unica tupla in presenza della clausola DISTINCT

Anagrafica Studenti

Matricola	*Nome*	*Cognome*	*DataNascita*
150	Alex	Del Piero	10/3/1975
151	Alex	Del Piero	11/2/2004
142	Giovanni	SenzaTerra	3/4/1250

$\pi_{matricola,nome}$ (studenti) DISTINCT

Nome	*Cognome*
Alex	Del Piero
Giovanni	SenzaTerra

Proiezione in SQL

SELECT [DISTINCT] lista_attributi FROM nome_tab

```
SELECT Nome, Cognome FROM ANAGRAFICA_STUDENTI
```

Nomi e cognomi (con ripetizioni) di tutti gli studenti presenti in anagrafica

```
SELECT DISTINCT Nome, Cognome FROM ANAGRAFICA_STUDENTI
```

Nomi e cognomi (senza ripetizioni) di tutti gli studenti presenti in anagrafica

Esempio

select Cognome from Studente

Matricola	Cognome	Nome
aaaa	Paolino	Paperino
bbbb	Rossi	Mario
cccc	Bianchi	Mario
dddd	Bianchi	Giuseppe
eeee	Rossi	Vincenzo
ffff	Esposito	Gennaro

Cognome
Paolino
Rossi
Bianchi
Bianchi
Rossi
Esposito

NON è una tabella

Esempio

- select * from Studenti

Matricola	Cognome	Nome
aaaa	Paolino	Paperino
bbbb	Rossi	Mario
cccc	Bianchi	Mario
dddd	Bianchi	Giuseppe
eeee	Rossi	Vincenzo
ffff	Esposito	Gennaro

Matricola	Cognome	Nome
aaaa	Paolino	Paperino
bbbb	Rossi	Mario
cccc	Bianchi	Mario
dddd	Bianchi	Giuseppe
eeee	Rossi	Vincenzo
ffff	Esposito	Gennaro

Selezione (1/2)

- L'operazione di selezione permette di scegliere tra le tuple di una relazione quelle che soddisfano una certa condizione logica

- Definizioni preliminari:
  - **Condizione Atomica** - Sia *r* una relazione su *R(X)* e siano $A \in X$ e $B \in X$ due attributi di *r*, e sia $\theta \in \{=, \neq, <, >, \leq, \geq\}$ una qualsiasi relazione definita sul dominio di *A* e *B*. Le seguenti sono condizioni atomiche: (i) $A \; \theta \; B$; (ii) $A = c$, per ogni valore costante *c* del dominio di *A*
  - **Condizione di Selezione** - Ogni condizione atomica è una Condizione di Selezione. D'altro canto, se *vc1* e *vc2* sono due condizioni di selezione, lo sono pure $vc1 \wedge vc2$, $vc1 \vee vc2$, e $\neg vc1$
  - **Soddisfacimento di una condizione di selezione** - Sia *R(X)* uno schema di relazione, *r* una istanza di *R*, *t* una tupla in r, *A* e *B* attributi in *X*, e *c* una costante. Sia χ una qualsiasi condizione di selezione su *A*, *B* e *c*. Il soddisfacimento di ζ da *t* ($t \not\models \chi$) è definita per induzione sulla struttura di χ, come segue:
    - Paragone tra attributi: $t \not\models A \; \theta \; B$ se e solo se $t[A] \; \theta \; t[B]$
    - Paragone tra costanti: $t \not\models A \; \theta \; c$ se e solo se $t[A] \; \theta \; c$

Selezione (2/2)

- Sia data una relazione *r* ϵ *R(X)* ed una condizione di selezione χ. Si Definisce **selezione** l'insieme:

$$\sigma_{\chi}(r) = \{t : t \models \chi\}$$

- Esempio

$\sigma_{Nome='Alex'}$(ANAGRAFICA_STUDENTI)

Matricola	*Nome*	*Cognome*	*DataNascita*
150	Alex	Del Piero	10/3/1975
151	Alex	Del Piero	11/2/2004

Selezione in SQL

SELECT [DISTINCT] lista_attributi FROM nome_tab
WHERE condizione

- Esempio

```
SELECT * FROM ANAGRAFICA_STUDENTI WHERE Nome='Alex'
```

studenti presenti in anagrafica con nome ‘Alex’

Esempio

- select *
  from Impiegato
  where Cognome='Bianchi'

Matricola	Cognome	Nome
aaaa	Paolino	Paperino
bbbb	Rossi	Mario
cccc	Bianchi	Mario
dddd	Bianchi	Giuseppe
eeee	Rossi	Vincenzo
ffff	Esposito	Gennaro

Matricola	Cognome	Nome
cccc	Bianchi	Mario
dddd	Bianchi	Giuseppe

Condizioni complesse

- È possibile usare connettivi logici (and, or, not) ed espressioni di confronto

```
select nome, cognome
from Impiegato where ufficio=20 and dipartimento='DIS'
```

```
select nome, cognome
from Impiegato where ufficio=20 or dipartimento='DIS'
```

```
select nome
from Impiegato where cognome = 'Rossi' and  (ufficio=20
or dipartimento='DIS')
```

Operatore LIKE

- È usato nel confronto di stringhe e si appoggia su due caratteri speciali:
  - _ (indica un qualsiasi carattere)
  - % (indica una stringa, anche vuota)

select nome
from Impiegato where cognome like ‘_o%i’

Ritorna tutti i nomi di Impiegato il cui cognome ha una “o” in seconda posizione e termina per “i” (Rossi, Doncelli, Loi)

Comportamento del valore NULL

and	V	U	F
V	V	U	F
U	U	U	F
F	F	F	F

Or	V	U	F
V	V	V	V
U	V	U	U
F	V	U	F

- U sta per UNKNOWN
- Per i valori nulli si usa
  - IS NULL
  - IS not NULL
- Esempi:

  WHERE stipendio IS NULL
  WHERE stipendio IS NOT NULL

Operatori di ordinamento

- Si possono ordinare le tuple risultato di un'interrogazione attraverso la clausola **ORDER BY** in maniera crescente (**ASC**) o decrescente (**DESC**)

```
SELECT Cognome, Nome
FROM ANAGRAFICA_STUDENTI
ORDER BY Cognome, Nome DESC
```

Ridenominazione

- Sia r una relazione definita su uno schema R(X) e sia Y un insieme di attributi avente la stessa cardinalità di X. Sia inoltre definito un ordinamento tra gli attributi di X

$$A_1 A_2 \ldots A_k$$

e quelli di Y

$$B_1 B_2 \ldots B_k$$

con la proprietà che

$$dom(A_i) = dom(B_i) \; \forall i = 1 \ldots k$$

- L'operazione di ridenominazione:

$$\varrho_{B_1 B_2 \ldots B_k \leftarrow A_1 A_2 \ldots A_k}(r) = \{t' : \exists t \in r \wedge t[A_i] = t'[B_i], \forall i = 1 \ldots k\}$$

Ridenominazione in SQL

- In SQL possiamo effettuare una ridenominazione attraverso il meccanismo degli *alias*.

  SELECT *A*1 **AS** *alias*1, *A*2 **AS** *alias*2 *: : : An* **AS** *aliasn*
  FROM *NOME TABELLA*

- Esempi:

  SELECT Nome **AS** Name, Cognome **AS** Surname
  FROM ANAGRAFICA DIPENDENTI;

  SELECT Nome Name, Cognome Surname
  FROM ANAGRAFICA DIPENDENTI;

  SELECT Nome **AS** “Nome dei dipendenti”
  FROM ANAGRAFICA DIPENDENTI

Ridenominazione in SQL

- select Cognome as Surname, Nome from Studente

Surname	Nome
Paolino	Paperino
Rossi	Mario
Bianchi	Mario
Bianchi	Giuseppe
Rossi	Vincenzo
Esposito	Gennaro

Estensione del SELECT

SELECT espressione(attributo) FROM nome_tab where condizione

- Esempio

  select cognome, Stipendio/12 as StipendioMensile
  from Impiegato
  where cognome=‘Rossi’

Definizione di Query SQL

- Una query SQL si ottiene combinando un'operazione di proiezione con una di selezione, potendo aggiungere la ridenominazione degli attributi

$$\pi_Y(\sigma_\chi(r))$$

SELECT Y **FROM** r **WHERE** χ

Dot Notation

- Qualora le due relazioni posseggano attributi con lo stesso nome si utilizza la cosiddetta *dot notation* per distinguere gli attributi comuni nel prodotto cartesiano

  NomeRelazione.NomeAttributo

- Esempi: r_1.nome, r_2.codice, etc…

Prodotto cartesiano

- Date due relazioni r_1 e r_2
  - definite sugli schemi $R(X_1)$ e $R(X_2)$
  - con $X_1 \cap X_2 = \emptyset$
    - (non hanno attributi in comune)

il **prodotto cartesiano** tra *r1* e *r2* restituisce un'istanza di relazione *r*

- definita su $R(X_1 \cup X_2)$
- contenente le tuple *t* formate dalla concatenazione di ciascuna tupla di r_1 con ciascuna tupla di r_2

$$r = r_1 \times r_2 = \{t = \langle t_1, t_2 \rangle, t_1 \in r_1 \wedge t_2 \in r_2\}$$

Esempio

Nome	*Cognome*
carlo	rossi
mirko	verdi
paolo	paolone

r_1

Codice	*Qualifica*
001	impiegato
002	dirigente

r_2

Nome	*Cognome*	*Codice*	*Qualifica*
carlo	rossi	001	impiegato
carlo	rossi	002	dirigente
mirko	verdi	001	impiegato
mirko	verdi	002	dirigente
paolo	paolone	001	impiegato
paolo	paolone	002	dirigente

$r_1 \times r_2$

Prodotto cartesiano SQL (modo implicito)

SELECT * FROM nome_tab$_1$, nome tab$_2$

Esempio: select * from dipendenti cross join mansioni;

```
1  select * from dipendenti, mansioni;
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO	CODMANSIONI	NOME
10	Carlo	Rosso	001	1000	001	impiegato
10	Carlo	Rosso	001	1000	002	dirigente
10	Carlo	Rosso	001	1000	003	segretario
11	Mirko	Verde	001	3000	001	impiegato
11	Mirko	Verde	001	3000	002	dirigente
11	Mirko	Verde	001	3000	003	segretario
12	Paolo	Nero	002	600	001	impiegato
12	Paolo	Nero	002	600	002	dirigente
12	Paolo	Nero	002	600	003	segretario

Download CSV

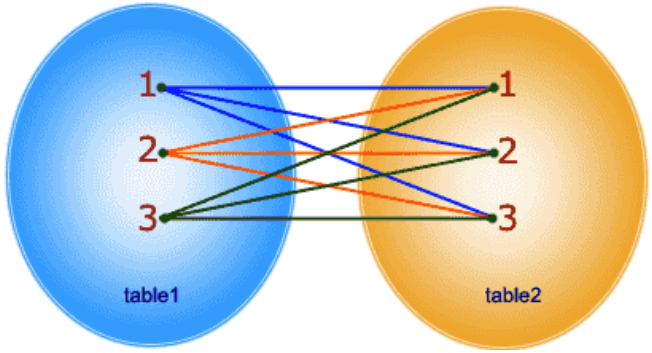
9 rows selected.

Prodotto cartesiano SQL (modo esplicito)

SELECT * FROM nome_tab$_1$ CROSS JOIN nome tab$_2$

Esempio:

select * from dipendenti cross join mansioni;



Si possono aggiungere condizioni:

select * from dipendenti cross join mansioni

where stipendio<1000;

Theta Join

- Date due relazioni r_1 su $R(X_1)$ ed r_2 su $R(X_2)$, con $X_1 \cap X_2 = \emptyset$, si definisce **THETA JOIN** l'operazione:

$$\sigma_\chi(r_1 \times r_2) \equiv r_1 \bowtie_\chi r_2$$

di selezione applicata al prodotto cartesiano

Theta Join in SQL (modo implicito)

SELECT * **FROM** r_1, r_2 **WHERE** *Condizione*

- Esempio

SELECT *
FROM nome_tab$_1$, nome_tab$_2$
WHERE (nome_tab$_1$.attributo$_1$ < 100
AND
nome_tab$_2$.attributo$_2$ = ‘STRINGA’)

Theta Join in SQL (modo esplicito)

SELECT * FROM *r1* **JOIN** *r2* **ON** *Condizione*

- Esempio

SELECT * FROM $nome\_tab_1$ **JOIN** $nome\_tab_2$

ON ($nome\_tab_1.attributo_1 < 100$

AND

$nome\_tab_2.attributo_2$ = 'STRINGA')

Equi Join

- Se la condizione di selezione χ è una condizione di uguaglianza tra un attributo di r_1 ed uno di r_2 si parla di **EQUI JOIN**

Nome	*Cognome*	*Mansione*
carlo	rossi	001
mirko	verdi	001
paolo	paolone	002

ANAGRAFICA_DIPENDENTI

Codice	*Qualifica*
001	impiegato
002	dirigente

MANSIONI

$$\text{ANAGRAFICA\_DIPENDENTI} \bowtie_{Codice=Mansione} \text{MANSIONI}$$

Nome	*Cognome*	*Mansione*	*Codice*	*Qualifica*
carlo	rossi	001	001	impiegato
mirko	verdi	001	001	impiegato
paolo	paolone	002	002	dirigente

Equi Join in SQL

- Modo implicito

```
SELECT *
FROM ANAGRAFICA_DIPENDENTI, MANSIONI
WHERE Mansione=Codice
```

- Modo esplicito

```
SELECT * FROM ANAGRAFICA_DIPENDENTI
                    JOIN
                    MANSIONI
ON Mansione=Codice
```

Ridenominazione tabelle

- Per evitare frasi lunghe quando serve la dot notation

```
SELECT  T1.attributox, T2.attributoy FROM
        ANAGRAFICA_DIPENDENTI   AS T1
        JOIN
        MANSIONI    AS T2
        ON T1.Mansione=T2.Codice
```

- **AS** può essere omesso

  FROM ANAGRAFICA_DIPENDENTI T1 **JOIN** MANSIONI T2

- La dot notation serve quando attributi appartenenti a tabelle diverse introducono ambiguità
- Per leggibilità usate Ti o TABi

Join Naturale

- Un caso particolare di theta join è il **JOIN NATURALE**, in cui la condizione χ è implicitamente verificata su *tutte le coppie di attributi delle due relazioni aventi lo stesso nome e si indica*

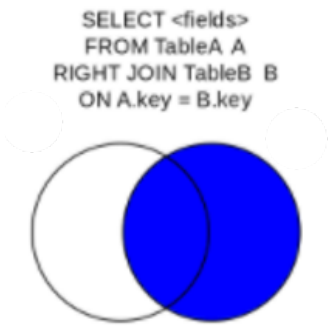
$$r_1 \bowtie r_2$$

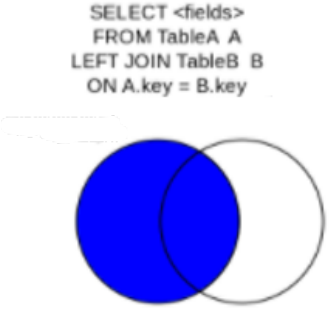
- Se, ad esempio, r_1 e r_2 hanno il nome l'attributo in comune A_c allora vale l'equivalenza

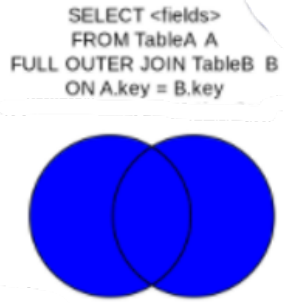
$$r_1 \bowtie r_2 \equiv r_1 \bowtie_{r_1.A_c=r_2.A_c} r_2$$

Join interno e esterno

- Esistono due tipi di JOIN
  - Interno o INNER
  - Esterno o OUTER







  - Le clausole INNER e OUTER sono facoltative

Tuple dondolanti

- La differenza tra Join e il prodotto Cartesiano è che il primo seleziona solo alcune tuple del secondo sulla base di una proprietà
- Le tuple che non sono contemplate nel Join vengono dette

dangling tuple

ANAGRAFICA_DIPENDENTI

Nome	*Cognome*	*Mansione*
carlo	rossi	001
mirko	verdi	001
paolo	paolone	002

MANSIONI

Codice	*Qualifica*
001	impiegato
002	dirigente
003	amministratore

- Nell'esempio tutte le tuple che si compongono con la terza tupla di Mansioni sono dondolanti

Considerazioni

- La presenza di tuple dangling che non partecipano alle relazioni di join può essere a volte dannosa in quanto alcune informazioni utili si perdono
- Per evitare ciò, esiste una variante dell'operazione di join detto **JOIN ESTERNO**

- In questo caso, date due relazioni *rs* e *rd*
  - dette anche relazione destra e relazione sinistra

  valgono le seguenti operazioni di esterno:
  - *join destro* $r_s \bowtie_{\text{RIGHT}} r_d$
  - *join sinistro* $r_s \bowtie_{\text{LEFT}} r_d$
  - *join completo* $r_s \bowtie_{\text{FULL}} r_d$

Join Esterno

- Il join esterno prevede che
  - tutte le tuple della relazione *rs* o della relazione *rd* o di entrambe concorrano alla formazione del risultato
  - ponendo NULL dove l'operazione di join coinvolge tuple dondolanti.

- In particolare nel:
  - join esterno sinistro vengono considerate tutte le tuple della relazione sinistra
  - join esterno destro vengono considerate tutte le tuple della relazione destra
  - join completo vengono considerate tutte le tuple della relazione sinistra e della relazione destra

RIGHT JOIN

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO
17	Angelo	Chianese	005	9000
10	Carlo	Rosso	001	1000
11	Mirko	Verde	001	3000
12	Paolo	Nero	002	600

rs

CODMANSIONI	NOME
001	impiegato
002	dirigente
003	segretario

rd

```
select * from dipendenti t1 right join mansioni t2
        on t1.mansioni=t2.codmansioni;
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO	CODMANSIONI	NOME
10	Carlo	Rosso	001	1000	001	impiegato
11	Mirko	Verde	001	3000	001	impiegato
12	Paolo	Nero	002	600	002	dirigente
-	-	-	-	-	003	segretario

LEFT JOIN

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO
17	Angelo	Chianese	005	9000
10	Carlo	Rosso	001	1000
11	Mirko	Verde	001	3000
12	Paolo	Nero	002	600

rs

CODMANSIONI	NOME
001	impiegato
002	dirigente
003	segretario

rd

```
select * from dipendenti t1 left join mansioni t2
        on t1.mansioni=t2.codmansioni;
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO	CODMANSIONI	NOME
17	Angelo	Chianese	005	9000	-	-
10	Carlo	Rosso	001	1000	001	impiegato
11	Mirko	Verde	001	3000	001	impiegato
12	Paolo	Nero	002	600	002	dirigente

FULL JOIN

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO
17	Angelo	Chianese	005	9000
10	Carlo	Rosso	001	1000
11	Mirko	Verde	001	3000
12	Paolo	Nero	002	600

rs

CODMANSIONI	NOME
001	impiegato
002	dirigente
003	segretario

rd

```
select * from dipendenti t1 full join mansioni t2
        on t1.mansioni=t2.codmansioni;
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO	CODMANSIONI	NOME
10	Carlo	Rosso	001	1000	001	impiegato
11	Mirko	Verde	001	3000	001	impiegato
12	Paolo	Nero	002	600	002	dirigente
-	-	-	-	-	003	segretario
17	Angelo	Chianese	005	9000	-	-

Vantaggi della ridenominazione

- L'uso di alias su variabile ed attributi permette di utilizzare più volte una stessa tabella in una interrogazione
  - Ad esempio se si vogliono trovare i nomi e cognomi dei dipendenti la cui mansione è uguale a quelle di un dipendente assegnato, allora è possibile usare la seguente query SQL

```
SELECT T1.Nome, T1.Cognome
FROM ANAGRAFICA DIPENDENTI T1, ANAGRAFICA DIPENDENTI T2
WHERE (T1.Mansione=T2.Mansione) AND (T2.Nome='carlo')
AND (T2.Cognome='rossi') AND (T1.Nome <> 'carlo') AND
(T1.Cognome <>'rossi')
```

Esercizi: creazione db di prova

```
create table dipendenti(
        coddipendenti    INT            primary key,
        nome             VARCHAR(30),
        cognome          VARCHAR(30),
        mansioni         CHARACTER(3),
        stipendio        int            default 0);

create table mansioni(
        codmansioni      CHARACTER(3)   primary key,
        nome             VARCHAR(30));

insert into dipendenti values (10,'Carlo','Rosso','001',1000);
insert into dipendenti values (11,'Mirko','Verde','001',3000);
insert into dipendenti values (12,'Paolo','Nero','002',600);

insert into mansioni values ('001','impiegato');
insert into mansioni values ('002','dirigente');
insert into mansioni values ('003','segretario');
```

Esercizi: query semplici da provare

```
select * from dipendenti;
select * from mansioni;
select mansioni from dipendenti;
select distinct mansioni from dipendenti;
select * from dipendenti where (stipendio < 1000);
select * from dipendenti where (cognome like 'R%');
select * from dipendenti order by cognome;
select cognome as Dipendente from dipendenti order by cognome;
select cognome as Dipendente, trunc(stipendio/12) as mensilita from dipendenti order by cognome;
```

Esercizi: query con join

```
select * from dipendenti, mansioni;
select * from dipendenti, mansioni
      where stipendio >= 1000 and mansioni.nome='dirigente';
select * from dipendenti join mansioni
      on stipendio >= 1000 and mansioni.nome='dirigente';
select * from dipendenti join mansioni
      on mansioni=codmansioni;
select t1.nome, t2.nome from dipendenti t1 join mansioni t2
      on t1.mansioni=t2.codmansioni;
select * from dipendenti t1 right join mansioni t2
      on t1.mansioni=t2.codmansioni;
select * from dipendenti t1 left join mansioni t2
      on t1.mansioni=t2.codmansioni;
select * from dipendenti t1 full join mansioni t2
      on t1.mansioni=t2.codmansioni;
```

Esercizi: complichiamo un pò

```
insert into dipendenti values (17,'Angelo','Chianese','005',9000);
select * from dipendenti;
select * from mansioni;

select * from dipendenti t1 right join mansioni t2
      on t1.mansioni=t2.codmansioni;
select * from dipendenti t1 left join mansioni t2
      on t1.mansioni=t2.codmansioni;
select * from dipendenti t1 full join mansioni t2
      on t1.mansioni=t2.codmansioni;
```

```
1  select * from dipendenti;
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO
10	Carlo	Rosso	001	1000
11	Mirko	Verde	001	3000
12	Paolo	Nero	002	600

Download CSV

3 rows selected.

```
1  select * from mansioni;
```

CODMANSIONI	NOME
001	impiegato
002	dirigente
003	segretario

Download CSV

3 rows selected.

```
1  select mansioni from dipendenti ;
```

MANSIONI
001
001
002

Download CSV

3 rows selected.

```
1  select distinct mansioni from dipendenti;
```

MANSIONI
002
001

Download CSV

2 rows selected.

```
1  select * from dipendenti where (stipendio < 1000);
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO
12	Paolo	Nero	002	600

Download CSV

```
1  select * from dipendenti where (cognome like 'R%');
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO
10	Carlo	Rosso	001	1000

Download CSV

```
1  select * from dipendenti order by cognome;
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO
12	Paolo	Nero	002	600
10	Carlo	Rosso	001	1000
11	Mirko	Verde	001	3000

Download CSV

3 rows selected.

```
1  select cognome as Dipendente from dipendenti order by cognome;
```

DIPENDENTE
Nero
Rosso
Verde

Download CSV

3 rows selected.

```
1  select cognome as Dipendente, trunc(stipendio/12) as mensilita
2  from dipendenti order by cognome;
```

DIPENDENTE	MENSILITA
Nero	50
Rosso	83
Verde	250

Download CSV

3 rows selected.

```
1  select * from dipendenti, mansioni;
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO	CODMANSIONI	NOME
10	Carlo	Rosso	001	1000	001	impiegato
10	Carlo	Rosso	001	1000	002	dirigente
10	Carlo	Rosso	001	1000	003	segretario
11	Mirko	Verde	001	3000	001	impiegato
11	Mirko	Verde	001	3000	002	dirigente
11	Mirko	Verde	001	3000	003	segretario
12	Paolo	Nero	002	600	001	impiegato
12	Paolo	Nero	002	600	002	dirigente
12	Paolo	Nero	002	600	003	segretario

Download CSV

9 rows selected.

```
1  select * from dipendenti, mansioni
2  where stipendio >= 1000 and mansioni.nome='dirigente';
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO	CODMANSIONI	NOME
10	Carlo	Rosso	001	1000	002	dirigente
11	Mirko	Verde	001	3000	002	dirigente

Download CSV

2 rows selected.

```
1  select * from dipendenti join mansioni
2          on mansioni=codmansioni;
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO	CODMANSIONI	NOME
10	Carlo	Rosso	001	1000	001	impiegato
11	Mirko	Verde	001	3000	001	impiegato
12	Paolo	Nero	002	600	002	dirigente

Download CSV

3 rows selected.

```
1  select t1.nome, t2.nome from dipendenti t1 join mansioni t2
2         on t1.mansioni=t2.codmansioni;
3
```

NOME	NOME
Carlo	impiegato
Mirko	impiegato
Paolo	dirigente

Download CSV

3 rows selected.

```
1  select * from dipendenti t1 right join mansioni t2
2         on t1.mansioni=t2.codmansioni;
3
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO	CODMANSIONI	NOME
10	Carlo	Rosso	001	1000	001	impiegato
11	Mirko	Verde	001	3000	001	impiegato
12	Paolo	Nero	002	600	002	dirigente
-	-	-	-	-	003	segretario

Download CSV

4 rows selected.

```
1  select * from dipendenti t1 left join mansioni t2
2         on t1.mansioni=t2.codmansioni;
3
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO	CODMANSIONI	NOME
10	Carlo	Rosso	001	1000	001	impiegato
11	Mirko	Verde	001	3000	001	impiegato
12	Paolo	Nero	002	600	002	dirigente

Download CSV

3 rows selected.

```
1  select * from dipendenti t1 full join mansioni t2
2         on t1.mansioni=t2.codmansioni;
3
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO	CODMANSIONI	NOME
10	Carlo	Rosso	001	1000	001	impiegato
11	Mirko	Verde	001	3000	001	impiegato
12	Paolo	Nero	002	600	002	dirigente
-	-	-	-	-	003	segretario

Download CSV

4 rows selected.

insert into dipendenti values (17,'Angelo','Chianese','005',9000);

```
1  select * from dipendenti;
2  select * from mansioni;
3
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO
17	Angelo	Chianese	005	9000
10	Carlo	Rosso	001	1000
11	Mirko	Verde	001	3000
12	Paolo	Nero	002	600

Download CSV

4 rows selected.

CODMANSIONI	NOME
001	impiegato
002	dirigente
003	segretario

Download CSV

3 rows selected.

```
1  select * from dipendenti t1 right join mansioni t2
2         on t1.mansioni=t2.codmansioni;
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO	CODMANSIONI	NOME
10	Carlo	Rosso	001	1000	001	impiegato
11	Mirko	Verde	001	3000	001	impiegato
12	Paolo	Nero	002	600	002	dirigente
-	-	-	-	-	003	segretario

Download CSV

4 rows selected.

```
1  select * from dipendenti t1 left join mansioni t2
2        on t1.mansioni=t2.codmansioni;
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO	CODMANSIONI	NOME
17	Angelo	Chianese	005	9000	-	-
10	Carlo	Rosso	001	1000	001	impiegato
11	Mirko	Verde	001	3000	001	impiegato
12	Paolo	Nero	002	600	002	dirigente

Download CSV

4 rows selected.

```
1  select * from dipendenti t1 full join mansioni t2
2         on t1.mansioni=t2.codmansioni;
```

CODDIPENDENTI	NOME	COGNOME	MANSIONI	STIPENDIO	CODMANSIONI	NOME
10	Carlo	Rosso	001	1000	001	impiegato
11	Mirko	Verde	001	3000	001	impiegato
12	Paolo	Nero	002	600	002	dirigente
-	-	-	-	-	003	segretario
17	Angelo	Chianese	005	9000	-	-

Download CSV

5 rows selected.