Algebra relazionale

Le basi di dati vengono utilizzate per rappresentare le informazioni di interesse per applicazioni che gestiscono dati. Conseguentemente i linguaggi per la specifica delle operazioni (di interrogazione e aggiornamento) sui dati stessi costituiscono una componente essenziale delle basi di dati e quindi di ciascun modello dei dati.
I linguaggi di interrogazione permettono il reperimento di dati da una base di dati.
Sono diversi dai linguaggi di programmazione:

- Non sono necessariamente Turing completi
- Non sono fatti per essere usati in calcoli complessi
- Supportano un accesso semplice ed efficiente a grandi insiemi di dati

Due linguaggi di interrogazione matematici formano la base per i linguaggi usati nella pratica (SQL):

- L'algebra relazionale
- Il calcolo relazionale

L'algebra relazionale è un linguaggio procedurale (in cui cioè le operazioni complesse vengono specificate descrivendo il procedimento da seguire per ottenere la soluzione).

Il calcolo relazionale (che non vedremo) è viceversa un **linguaggio dichiarativo**, in cui le espressioni descrivono le proprietà del risultato, piuttosto che la procedura per ottenerlo. Questo linguaggio è basato sul calcolo dei predicati del primo ordine.

I due precedenti linguaggi **non** sono utilizzati dall'utente di un sistema di database. **Il linguaggio di riferimento per i database relazionali è SQL** che combina gli aspetti dichiarativi del calcolo e quelli procedurali dell'algebra.
In particolare SQL è un linguaggio dichiarativo nell'aspetto di interrogazione dei dati.
L'interrogazione SQL per essere eseguita viene passata all'ottimizzatore di interrogazioni (query optimizer), un componente del DBMS che analizza l'interrogazione e formula a partire da questa un'interrogazione equivalente nel linguaggio procedurale interno del sistema di gestione di basi di dati. Questo linguaggio procedurale è nascosto all'utente.

Cenni di algebra relazionale

L'algebra relazionale NON è un linguaggio usato dagli utenti dei sistemi di basi di dati. È un linguaggio procedurale basato su concetti di tipo algebrico.
Nell'algebra relazionale le operazioni complesse vengono specificate descrivendo il procedimento da seguire per ottenere la soluzione. L'algebra relazionale è costituita da operatori che:

- Sono definiti su **relazioni**
- Producono una **relazione** come risultato
- Possono essere composti per formare interrogazioni complesse.

Gli operatori dell'algebra relazionale sono:

- Insiemistici: unione, intersezione, differenza, prodotto cartesiano
- Operatori specifici: ridenominazione, selezione, proiezione
- L'operatore di **join**, che può essere:
    - **Inner join** (nelle forma di theta join, **equi-join,** natural join,)
    - **Outer join** (left, right o full outer join).

Operazioni insiemistiche

Le relazioni sono insiemi di tuple omogenee (due tuple della relazione sono definite sugli stessi attributi).
In quanto **insiemi**, ha quindi senso applicare loro le operazioni tipiche degli insiemi: unione, intersezione, differenza.
In quanto insiemi di **tuple omogenee**, però, le operazioni hanno significato solo se applicate a relazioni definite sugli stessi attributi. Solo in questo modo, il risultato sarà ancora una relazione costituita da tuple omogenee, definita sugli stessi attributi delle relazioni di partenza.

- L'unione di due relazioni R1 ed R2 definite sullo stesso insieme di attributi X è ancora definita su X che contiene le tuple che appartengono a R1 oppure a R2 oppure ad entrambe.
- L'intersezione di R1 e R2 è una relazione le cui tuple appartengono sia a R1 che a R2
- La differenza R1 – R2 è una relazione contenente le tuple che appartengono a R1 e non appartengono a R2

LAUREATI	
Matricola	Nome
1	Mario
2	Merio
3	Mirio

DIRIGENTI	
Matricola	Nome
3	Mirio
2	Merio
12	Oddone

LAUREATI U IRIGENTI	
Matricola	Nome
1	Mario
2	Merio
3	Mirio
12	Oddone

LAUREATI ∩ DIRIGENTI	
Matricola	Nome
2	Merio
3	Mirio

LAUREATI - DIRIGENTI	
Matricola	Nome
1	Mario

Possono esserci situazioni in cui ha significato compiere operazioni insiemistiche tra relazioni che hanno attributi di nome diverso, ma dello stesso tipo. Due relazioni si dicono **compatibili rispetto all'unione** se i loro attributi corrispondano nel numero e nel tipo (anche se non nel nome). Si possono allora ridenominare gli attributi per rientrare nel caso precedente (vedi operazione di ridenominazione) e procedere con l'operazione insiemistica. Conseguentemente diremo che la condizione per eseguire operazioni insiemistiche tra due relazioni è la compatibilità rispetto all'unione.

Nota su SQL: Il linguaggio SQL definisce le operazioni insiemistiche con gli operatori: UNION, INTERSECT, MINUS (o EXCEPT) ma non tutti i sistemi commerciali offrono una implementazioni di queste funzionalità. In generale però implementano l'operazione di UNION (le altre operazioni sono comunque ottenibili per via indiretta).
Nel caso di SQL, non è richiesto come in algebra relazionale che i nomi degli attributi siano uguali ma solo che le relazioni siano compatibili rispetto all'unione. La relazione risultante avrà come nomi degli attributi quelli del primo operando. IN SQL l'unione si fa con l'operatore:

- **UNION**: la tabella risultante contiene tutti i record della prima e tutti i record della seconda tabella. Eventuali duplicati sono rimossi.

Vediamo ora le operazioni di prodotto cartesiano e le operazioni specifiche dell'algebra relazionale: ridenominazione, selezione e proiezione. Tutte queste operazioni saranno realizzate in SQL con **la sola istruzione *select***, opportunamente adeguata allo specifico scopo.

Ridenominazione: operatore ρ

L'operatore di ridenominazione modifica lo schema di una relazione cambiando il nome di uno o più attributi

PATERNITÀ	
Padre	Figlio
Dino	Mario
Mino	Merio
Tino	Mirio

$\rho\,((Padre \rightarrow Genitore), PATERNITÀ)$

PATERNITÀ	
Genitore	Figlio
Dino	Mario
Mino	Merio
Tino	Mirio

- In SQL: *select Padre* ***as Genitore****, Figlio from PATERNITA*

Prodotto cartesiano: operatore x

È un operatore insiemistico. Per questa operazione non è richiesto che le relazioni di partenza siano compatibili rispetto all'unione (nessuna restrizione sulle relazioni di partenza).

Data due relazioni, R, S, il loro prodotto cartesiano si indica con R x S e restituisce una relazione il cui **schema** contiene tutti gli attributi di R seguiti da tutti gli attributi di S nell'ordine originale:

AT_1	AT_2	ATT_3

X

AT_A	AT_B

=

AT_1	TT_2	AT_3	T_A	ATT_B

L'istanza della relazione risultante, cioè l'istanza del prodotto cartesiano, contiene una tupla <r,s> per ogni tupla r in R ed ogni tupla s in S. Quindi le tuple di R x S sono costituite da tutte le possibili concatenazioni delle tuple R con le tuple di S.

IMPIEGATO	
Nome	**Matricola**
Dino	**1**
Mino	2
Tino	3

DIPARTIMENTO	
Codice	**Denominazione**
X	SIA
Y	AFM

IMPIEGATO x DIPARTIMENTO			
Nome	Matricola	**Codice**	**Denominazione**
Dino	**1**	**X**	**SIA**
Dino	**1**	**Y**	**AFM**
Mino	2	X	SIA
Mino	2	Y	AFM
Tino	1	X	SIA
Tino	2	Y	AFM

FRUTTA	
Cod	**Descr**
a	**Mela**
b	Pera

NUMERI_ROMANI x FRUTTA			
Simbolo	**NUMERI_ROMANI.Descr**	**Cod**	**FRUTTA.Descr**
I	**1**	**a**	**Mela**
I	1	b	Pera
II	**2**	**a**	**Mela**
II	2	b	Pera
V	**5**	**a**	**Mela**
V	5	b	Pera

Selezione: operatore σ_P

L'operatore di selezione estrae le sole righe di una relazione che soddisfano una condizione P (predicato, ovvero una espressione booleana).

Data una relazione R ed una condizione P, la selezione di R per la condizione P è una relazione costituita dagli stessi attributi di R che contiene tutte le tuple di R che soddisfanno la condizione P.

IMPIEGATO	
Nome	Matricola
Dino	**1**
Rino	2
Tino	3

$\sigma_{Nome='Dino'}(IMPIEGATO)$ →

Nome	Matricola
Dino	**1**

$\sigma_{Nome='Dino' \text{ AND } Matricola = 2}(IMPIEGATO)$ →

Nome	Matricola

ø

- In SQL: *select Nome, Matricola from IMPIEGATO* ***where Nome = 'Dino'***

Proiezione: operatore Π_A

L'operatore di **proiezione** estrae dalla relazione le colonne di interesse.
Data una relazione R ed un insieme di attributi A = {att1, att2, att3} la proiezione di R su A è una relazione il cui schema contiene solo gli attributi di A e la cui istanza contiene tutte le tuple di R ristrette ad A.

IMPIEGATO			
Cod	Cognome	Nome	Data di nascita
12	Arro	Tino	1/1/1960
13	Manda	Rino	1/1/1960
14	Accen	Dino	1/1/1960
15	Panno	Lino	1/1/1960

$\Pi_{Cod,\ Nome}(IMPIEGATO)$ →

Cod	Nome
12	Tino
13	Rino
14	Dino
15	Lino

- In SQL: ***select Cod, Nome from*** *Impiegato*

Combinare selezione e proiezione

Selezione e proiezione sono entrambe definite su un solo operando, una relazione, e come risultato **producono una porzione dell'operando.**
La selezione produce **un sottoinsieme** delle tuple **su tutti gli attributi** (decomposizione orizzontale).
La proiezione produce un risultato cui contribuiscono tutte le tuple, ma su un **sottoinsieme di attributi** (decomposizione verticale).

Siccome gli operatori dell'algebra relazionale si applicano a relazioni per produrre relazioni, è possibile applicare gli operatori in successione: l'operatore che segue è applicato alla relazione prodotta dall'operatore precedente.

IMPIEGATO			
Cod	Cognome	Nome	Data di nascita
12	Rossi	Tino	1/1/1960
13	Neri	Rino	1/1/1960
14	Bianchi	Dino	1/1/1960
15	Verdi	**Lino**	1/1/1960

$\Pi_{Nome}(\sigma_{Cod=15}(IMPIEGATO))$

Nome
Lino

- In SQL: **select Nome from** impiegato **where Cod = 15**

Predicati sui valori NULL

NULL non è uguale ad alcun valore. Il predicato 5 = NULL non è ne true né False, ma **UNKNOWN.**
Anche l'espressione booleana NULL = NULL dà risultato dà risultato **UNKNOWN**: sono due informazioni mancanti, non si può stabilire l'uguaglianza.

PROGETTO			
Progetto	**Inizio**	**Durata**	**Costo**
ALPHA	1/1/1999	12	NULL
BETA	2/2/2010	21	800
GAMMA	3/3/2019	8	100

Nella relazione in esempio, la seguente: $\sigma_{Costo=NULL}(PROGETTO)$
Restituisce come risultato l'insieme vuoto.
D'altra parte, la seguente: $\sigma_{Costo > 100}(PROGETTO)$
Non prende in considerazione il progetto per ALPHA per il quale L'attributo Costo vale NULL, non essendo possibile istituire alcun confronto. Per selezionare le tuple di una relazione in cui un attributo assume valore NULLL, si deve usare uno speciale predicato: IS NULL. Esiste anche il predicato IS NOT NULL.

Esempio

$\sigma_{Costo\ IS\ NULL}(PROGETTO)$			
Progetto	**Inizio**	**Durata**	**Costo**
ALPHA	1/1/1999	12	NULL

$\sigma_{Costo\ IS\ NOT\ NULL}(PROGETTO)$			
Progetto	**Inizio**	**Durata**	**Costo**
BETA	2/2/2010	21	800
GAMMA	3/3/2019	8	100

ESERCIZIO:
Data la seguente relazione, che elenca i dipendenti indicando per ciascuno il codice del dirigente cui fa capo, come si trova in algebra relazionale l'elenco dei dipendenti senza dirigente?

IMPIEGATO			
Codice	**Cognome**	**Stipendio**	**Dir**
1	Bianchi	26000	5
2	Neri	24000	5
3	Rossi	28000	5
4	Viola	30000	5
5	Gatti	40000	6
6	Leoni	45000	NULL
7	Mori	24000	6

Join: ⋈

(codice unicode: 8904)

L'operatore di join è il più caratteristico dell'algebra relazionale: **permette di correlare dati contenuti in relazioni diverse, confrontando i valori contenuti in esse** e utilizzando quindi la caratteristica fondamentale del modello, quella di essere basato su valori. Esistono diverse forme dell'operatore di Join, la più rilevante dal punto di vista pratico è il l'**equi-join**, una variante del theta join (o join condizionale).

L'operatore di Join è un operatore derivato: equivale all'operazione di **prodotto cartesiano tra due relazioni seguito da una selezione**.

Def: **Il Join di due relazioni R ed S è un sottoinsieme del loro prodotto cartesiano, costituito dalla concatenazione delle sole tuple di R ed S che soddisfano una proprietà (condizione di join).**

L'operazione di join tra due relazioni R(A_1, A_2,…,A_n) ed S(B_1, B_2,…,B_M) è indicata al seguente modo:

$R \bowtie_p S$, in cui p (predicato) è la condizione di join in forma di espressione booleana

Di seguito vengono esposti tre tipi di join che ricadono tutti nella categoria di **inner join** (Join interni)

- Theta join
- Equi join (MOLTO IMPORTANTE)
- Natural join

Theta Join: $R \bowtie_p S$

Nel **Theta join, o join condizionale**, la condizione di join tra due relazioni R ed S è una espressione booleana composta con il connettivo logico AND da espressioni booleane semplici della forma $A_i \, \theta \, B_j$ dove A_i è un attributo di R, B_j è un attributo di S, e θ (theta) è un operatore di confronto: { =, <, ≤ , >, ≥, ≠}

Esempio:$R \bowtie_{Id=Cod \text{ AND } S.Descr = R.Descr} S$

- In Sql: **select** listaAttributi **from R join S on** Id = Cod AND R.Descr = S.Descr

Per semplicità di scrittura, indicheremo l'operazione di join seguente modo: **R join S on p** in linea con la sintassi SQL

Equi-join

Caso particolare del Theta Join **è l'equi join**, in cui la condizione di join p è composta **solo da uguaglianze** (eventualmente connesse con l'AND). È un caso molto importante perché la maggior parte delle interrogazioni reali sono di questo tipo.

IMPIEGATO		
Matricola	Cognome	CodProgetto
12	Rossi	ALPHA
13	Neri	ALPHA
14	Bianchi	GAMMA
15	**Verdi**	**NULL**

PROGETTO			
Cod	Inizio	Durata	Costo
ALPHA	1/1/1999	12	400
BETA	2/2/2010	21	800
GAMMA	3/3/2019	8	100

IMPIEGATO join PROGETTO on CodProgetto = Cod						
Matricola	Cognome	CodProgetto	Cod	Inizio	Durata	Costo
12	Rossi	ALPHA	ALPHA	1/1/1999	12	400
13	Neri	ALPHA	ALPHA	1/1/1999	12	400
14	Bianchi	GAMMA	GAMMA	3/3/2019	8	100

Osserva attentamente che le tuple delle relazioni di partenza in cui gli attributi di join valgono NULL, non compiano nel risultato.

Osserva che nel join compaiono gli attributi CodProgetto e Cod che contengono gli stessi valori poiché usati nella condizione di equi join: solo le tuple di IMPIEGATO e PROGETTO con uguale valore in questi attributi sono state concatenate. Possiamo rimuoverne uno con una operazione di Proiezione:

Π_A(IMPIEGATO **join** PROGETTO **on** CodProgetto = Cod **)**, **A** = {Matricola,Cognome,Progetto,Inizio,Durata,Costo}

- In Sql: *select Matricola, Cognome, Progetto, Inizio, Durata, Costo* ***from IMPIEGATO join PROGETTO on CodProgetto = Cod***

Se fossimo interessati a conoscere solo i dettagli degli impiegati impegnati in qualche progetto e il costo del progetti, per i soli progetti di costo maggiore a 100, dovremo eseguire la successione di tre operazioni:

- Un **equi join** tra le due tabelle per mettere in corrispondenza impiegati e progetti
- Una **selezione**, per estrarre solo le tuple del join precedente, per le quali il costo dei progetti è maggiore di 100
- Una **proiezione** per mostrare solo gli attributi di interesse.

In algebra relazionale questa interrogazione si scrive così:
Π_A ($\sigma_{costo > 100}$ (IMPIEGATO join PROGETTO on CodProgetto = Cod)) , A= {Matricola, Cognome, Cod, Costo}

In Sql, la stessa interrogazione la scriveremo così:
select Matricola, Cognome, Cod, Costo from Impiegato join Progetto on CodProgetto = Cod where costo > 100

Natural Join: ⋈

Nel caso in cui le relazioni su cui si opera avessero attributi con lo stesso nome, si può eseguire una operazione di natural join che è una operazione in cui sono combinate le sole tuple di due relazioni che hanno valori uguali in attributi con lo stesso nome. Si indica al seguente modo: R ⋈ S
Nel caso del natural join la condizione di join è implicita: valori uguali su attributi con lo stesso nome.
La relazione risultante ha per attributi l'unione degli attributi.

IMPIEGATO		
Matricola	Cognome	Progetto
12	Rossi	ALPHA
13	Neri	ALPHA
14	Bianchi	GAMMA
15	Verdi	NULL

PROGETTO			
Progetto	Inizio	Durata	Costo
ALPHA	1/1/1999	12	400
BETA	2/2/2010	21	800
GAMMA	3/3/2019	8	100

IMPIEGATO ⋈ PROGETTO					
Matricola	Cognome	Progetto	Inizio	Durata	Costo
12	Rossi	ALPHA	1/1/1999	12	400
13	Neri	ALPHA	1/1/1999	12	400
14	Bianchi	GAMMA	3/3/2019	8	100

Osserva che nel natural join l'attributo presente in entrambe le relazioni di partenza su cui viene effettuato il join, non viene ripetuto (stesso nome in entrambe le tabelle)

- In SQL: *select Matricola, Cognome, IMPIEGATO.Progetto, Inizio, Durata, Costo from* ***IMPIEGATO natural join PROGETTO***
- La condizione di join è implicita: l'uguaglianza su attributi con lo stesso nome
- La maggior parte dei DBMS effettivamente esistenti non usa il natural join, che si basa sui nomi degli attributi, ma il theta join e ancor più l'equi join.

Outer join

Le operazioni di inner join producono come risultato una relazione alla cui istanza contribuiscono solo le tuple delle due relazioni di partenza che corrispondono secondo un criterio dato. Per esempio nel natural join e nell'equi join, compaiono nel risultato solo le righe per le quali i campi corrispondenti delle due relazioni hanno lo stesso valore, mentre le tuple di una relazione che non hanno una controparte nell'altra, sono tralasciate dall'operazione di join.
In un esempio precedente:

IMPIEGATO		
Matricola	Cognome	CodProg
12	Rossi	ALPHA
13	Neri	ALPHA
14	Bianchi	GAMMA
15	**Verdi**	**NULL**

IMPIEGATO join PROGETTO on CodProg = Cod						
Matricola	Cognome	CodProg	Cod	Inizio	Durata	Costo
12	Rossi	ALPHA	ALPHA	1/1/1999	12	400
12	Rossi	ALPHA	ALPHA	1/1/1999	12	400
14	Bianchi	GAMMA	GAMMA	3/3/2019	8	100

La tupla relativa all'IMPIEGATO Verdi, che presenta **NULL** nell'attributo Progetto, non può comparire nel risultato di join che si basa sull'uguaglianza di valori negli attributi specificati dalla condizione di join: CodProg = Cod. In generale solo un **sottoinsieme** delle tuple di ciascuna tabella avranno una controparte nell'altra. Come conseguenza, esisteranno tuple della prima relazione e tuple della seconda relazione che non compariranno nel risultato finale.

Nella pratica esistono diverse situazioni in cui si preferisce **mantenere tutte** le tuple di una o dell'altra tabella, anche quelle che non hanno una controparte corrispondente. Nell'esempio precedente, si potrebbe voler conoscere i dettagli degli impiegati e dei progetti in cui sono impegnati, includendo anche eventualmente le informazioni degli impiegati che non dovessero essere impegnati in alcun progetto (Verdi). Esiste questo scopo un'altra forma di join, il join esterno, molto **importante** nella pratica, che prevede che **tutte** le tuple di una o dell'all'atra relazione diano un contributo al risultato, anche quelle che non hanno una opportuna controparte nell'altra tabella ed in questo caso le tuple di origine vengono estese con valori null.

Join esterno….						
Matricola	Cognome	CodProg	Cod	Inizio	Durata	Costo
12	Rossi	ALPHA	ALPHA	1/1/1999	12	400
12	Rossi	ALPHA	ALPHA	1/1/1999	12	400
14	Bianchi	GAMMA	GAMMA	3/3/2019	8	100
15	Verdi	NULL	**NULL**	**NULL**	**NULL**	**NULL**

L'outer join nella sua definizione formale estende il natural join che combina le relazioni sulla base di valori uguali con attributi con lo stesso nome. Nella realtà però gli outer join esistono e vengono realizzati come estensione di un **theta join**, o join condizionale, sulla base di una condizione specificata. Siccome nella maggior parte dei casi il join che si realizza è l'equi-join, l'outer join si realizza per lo più come equi-join esteso.
Faremo riferimento al theta join come punto di partenza dell'outer join.
L'outer join si presenta in tre varianti: left, right e full outer join.

Left outer Join

Il left join tra due tabelle R ed S, che indicheremo come R left join S on P, estende un inner join includendo anche tutte le tuple della prima tabella che non soddisfano la condizione di join, cioè che non hanno un controparte nella seconda relazione tale che la condizione di join sia soddisfatta, completando le tuple del risultato con valori NULL.
Quindi il risultato del left join include sicuramente **tutte le tuple della prima relazione**. Tra queste, congiunge con le righe della seconda relazione quelle che hanno una controparte tale da soddisfare la condizione di join, mentre le tuple della prima relazione che non hanno controparte valida nella seconda, sono **completate con valori NULL**.

- In SQL: *select listaAttributi from Impiegato* ***left join*** *Progetto* ***on*** *CodProg = Cod*

IMPIEGATO **left outer join** PROGETTO **on** CodProg = Cod						
Matricola	Cognome	CodProg	Cod	Inizio	Durata	Costo
12	Rossi	ALPHA	ALPHA	1/1/1999	12	400
12	Rossi	ALPHA	ALPHA	1/1/1999	12	400
14	Bianchi	GAMMA	GAMMA	3/3/2019	8	100
15	Verdi	NULL	**NULL**	**NULL**	**NULL**	**NULL**

Right Outer join

Analogamente al precedente, nel right outer join – indicato come **R right join S on P** - sono presenti tutte le tuple della seconda relazione. Di queste, le tuple che hanno una controparte valida nella prima relazione, sono ad esse concatenate. Quelle che non hanno una controparte valida nella prima relazione, sono estese con valori NULL.

- In SQL: *select listaAttributi from Impiegato* ***right join*** *Progetto* ***on*** *CodProg = Cod*

Full outer join

Il full outer join - R full join S on P - combina le caratteristiche del left e del right outer join, includendo tutte le tuple della prima relazione, tutte le tuple della seconda relazione, concatenando quelle che soddisfano la condizione di join ed estendendo con valori NULL quelle che non hanno una controparte nell'altra tabella.

- In SQL: *select listaAttributi from Impiegato* ***full join*** *Progetto* ***on*** *CodProg = Cod*

IMPIEGATO	
Impiegato	CodRep
Rossi	amministrazione
Neri	vendite
Bianchi	vendite

UFFICIO	
Cod	Capo
produzione	**Pesca**
vendite	**Mela**

IMPIEGATO join UFFICIO on CodRep = Cod			
Impiegato	CodRep	Cod	Capo
Neri	vendite	vendite	Mela
Bianchi	vendite	vendite	Mela

IMPIEGATO left join UFFICIO on CodRep = Cod			
Impiegato	CodRep	Cod	Capo
Rossi	amministrazione	**NULL**	**NULL**
Neri	vendite	vendite	Mela
Bianchi	vendite	vendite	Mela

IMPIEGATO right join UFFICIO on CodRep = Cod			
Impiegato	CodRep	Cod	Capo
NULL	**NULL**	produzione	**Pesca**
Neri	vendite	vendite	**Mela**
Bianchi	vendite	vendite	**Mela**

IMPIEGATO full join UFFICIO on CodRep = Cod			
Impiegato	CodRep	Cod	Capo
Rossi	amministrazione	**NULL**	**NULL**
Neri	vendite	vendite	**Mela**
Bianchi	vendite	vendite	**Mela**
NULL	**NULL**	produzione	**Pesca**

Caso particolare: il self join

È possibile mettere in join una tabella con sé stessa. Questa situazione è trattata in modo identico alle altre, ma vista la particolarità del caso, si parla di self join (che non è un'altra operazione: riguarda il fatto che i due operandi del join sono la stessa tabella).

IMPIEGATO			
Codice	**Cognome**	**Stipendio**	**Dir**
1	Bianchi	26000	5
2	Neri	24000	5
3	Rossi	28000	5
4	Viola	30000	5
5	Gatti	40000	6
6	Leoni	45000	NULL
7	Mori	24000	6

Consideriamo ad esempio la relazione di schema:
IMPIEGATO(Codice, Cognome, Stipendio, Dir)
Nella relaziuone sono elencati i dipendenti di una azienda con l'indicazione del codice del rispettivo dirigente, che è anch'egli un impiegato. L'attributo Dir, che contiene il valore del codice del dirigente a capo di un impiegato, è chiave esterna sull'attributo codice della stessa tabella IMPIEGATO. Bianchi ha come dirigente Gatti, mentre Gatti non ha dirigente. Volendo elencare tutti gli impiegati con i dati anagrafici dei rispettivi dirigenti, bisogna eseguire un left outer join la tabella IMPIEGATO e se stessa. Per evitare ambiguità, dobbiamo prima applicare l'operare di ridenominazione per ottenere una relazione uguale a impiegati ma con attributi di diverso nome. Modifichiamo il nome degli attributi anteponendo il prefisso Dir e chiamiamo DIR_IMPIEGATO questa nuova relazione:
IMPIEGATO **left join** DIR_IMPIEGATO on Dir = DirCodice

IMPIEGATO left join DIR_IMPIEGATO on Dir = DirCodice							
Codice	**Cognome**	**Stipendio**	**Dir**	**DirCodice**	**DirCognome**	**DirStipendio**	**DirDir**
1	Bianchi	26000	5	5	Gatti	40000	6
2	Neri	24000	5	5	Gatti	40000	6
3	Rossi	28000	5	5	Gatti	40000	6
4	Viola	30000	5	5	Gatti	40000	6
5	Gatti	40000	6	6	Leoni	45000	NULL
6	Leoni	45000	NULL	**NULL**	**NULL**	**NULL**	**NULL**
7	Mori	24000	6	6	Leoni	45000	NULL

Con una proiezione, elenchiamo solo gli attributi di interesse:

Codice	**Cognome**	**Stipendio**	**DirCognome**
1	Bianchi	26000	Gatti
2	Neri	24000	Gatti
3	Rossi	28000	Gatti
4	Viola	30000	Gatti
5	Gatti	40000	Leoni
6	Leoni	45000	**NULL**
7	Mori	24000	Leoni

Nota che se non avessimo fatto un left join, non avremmo elencato l'impiegato Leoni che non ha dirigente.

Se volessimo conoscere quali dipendenti non hanno dirigente

Esercizio Svolto: verifiche degli studenti

Consideriamo lo schema di database:
STUDENTE(Matricola, Nome, Cognome, Indirizzo, Telefono)
MATERIA(Codice, NomeMateria, NumeroOre)
PROVA (ID, MatricolaStud, CodiceMat, Data, Voto)

Si hanno le seguenti foreign key:
PROVA (MatricolaStud) ⊆ STUDENTE(Matricola)
PROVA (CodiceMat) ⊆ MATERIA (Codice)

Elenca le materie insegnate per più di 3 ore settimanali:
π_A ($\sigma_{\text{NumeroOre >= 3}}$ (MATERIA)), A= { NomeMateria, NumeroOre}

Elenca nome, cognome e voto degli studenti interrogati in una materia di cui si conosce il codice da indicare con: **[materia scelta]**:
π_A ($\sigma_{\text{CodiceMat = [materia scelta]}}$ (STUDENTE join PROVA on Matricola = MatricolaStud)) , A= {Nome, Cognome, Voto}

Elenca NomeMateria, Data, Voto, per le prove di uno studente con voto non inferiore a 6, conoscendo la matricola dello studente **[matricola scelta]**:
π_A ($\sigma_{\text{MatricolaStud = [matricola scelta] AND Voto >= 6}}$ (PROVA) join MATERIA on CodMateria = Codice),
A= { NomeMateria, Data, Voto}

Elenca Nome, Cognome, Matricola degli studenti senza voti
π_A ($\sigma_{\text{MatricolaStud IS NULL}}$ (STUDENTE left join PROVA on Matricola = MatricolaStud)) , A= {Nome, Cognome, Matricola}

Elenca Cognome, Nome, Data, Voto degli studenti interrogati in una materia di cui si conosce il nome
π_A (STUDENTE **join** ($\sigma_{\text{NomeMateria = [materia scelta]}}$ (MATERIA) join PROVA on Codice = CodiceMat) **on** Matricola = MatricolaStud), A= {Cognome, Nome, Data, Voto}

Esercizio Svolto: Distribuzione commerciale di una azienda

Consideriamo il seguente schema di database per rappresentare le informazioni su Agenti di commercio e Clienti. Ogni agente gestisce uno o più clienti, mentre ogni cliente è gestito da un solo agente.

Nello schema del database:
AGENTE(IDAgente, Nome, Cognome)
CLIENTE(IDCliente, RagioneSoc, PartitaIVA, Provincia, IDAgente)

I vincoli di chiave e di chiave esterna che potremmo ragionevolmente porre sono:
AGENTE
primary Key : IDAgente

CLIENTE
primary Key : IDCliente
foreign key : CLIENTE (IDAgente) ⊆ AGENTE (IDAgente)

Consideriamo la seguente possibile istanza del database:
Si osservi che questa istanza di database violerebbe il vincolo di integrità referenziale indicato se fosse imposto: CLIENTE Prome ha come agente di vendtia *Gia* che non compare nella relazione AGENTE. In presenza di un vincolo di integrità referenziale il DBMS ne avrebbe impedito l'inserimento.

CLIENTE				
IDCliente	RagioneSoc	PartitaIVA	Provincia	IDAgente
Lami	Lamiere per auto	04357839912	TO	Bia
Levi	Levigatoria Toscana	01357739913	FI	Ner
Luci	Lucidatura Metalli	02357639914	RM	Ner
Meta	Metallurgica TS	03357539916	BO	Ner
Metb	Metalli Rari	05357439917	NA	Ros
Otto	Ottonifico BA	06357399181	BA	Ros
Prome	Prodotti Metallici	07357239919	BG	Gia
Rame	Rame & Metalli	08357139932	PA	Ver
Tond	Tondini metallici	09357039933	BS	Bia
Vite	Viteria MI	01157939944	MI	Bia
Vitp	Viteria di precisione	01257839955	MI	Bia

AGENTE		
IDAgente	Nome	Zona
Bia	Bianchi	Mord
Bru	Bruni	Centro
Ner	Neri	Centro
Ros	Rossi	Sud
Ver	Verdi	Isole

Calcoliamo la selezione di Agenti per la zona centro: $\sigma_{Zona="Centro"}(AGENTE)$

IDAgente	Nome	Zona
Bru	Bruni	Centro
Ner	Neri	Centro

Calcoliamo la proiezione di Agenti su Nome, Zona: $\Pi_{Nome,\ Zona}(AGENTE)$

Nome	Zona
Bianchi	Mord
Bruni	Centro
Neri	Centro
Rossi	Sud
Verdi	Isole

Calcoliamo: $\sigma_{Provincia = "MI"}(Cliente)$

IDCliente	RagioneSociale	PartitaIVA	Provincia	IDAgente
Vite	Viteria MI	01157939944	MI	Bia
Vitp	Viteria di precisione	01257839955	MI	Bia

Calcoliamo: $\Pi_{IDCliente,\ RagioneSociale,\ IDAgente}(CLIENTE)$

IDCliente	RagioneSociale	IDAgente
Lami	Lamiere per auto	Bia
Levi	Levigatoria Toscana	Ner
Luci	Lucidatura Metalli	Ner
Meta	Metallurgica TS	Ner
Metb	Metalli Rari	Ros
Otto	Ottonifico BA	Ros
Prome	Prodotti Metallici	Gia
Rame	Rame & Metalli	Ver
Tond	Tondini metallici	Bia
Vite	Viteria MI	Bia
Vitp	Viteria di precisione	Bia

AGENTE		
IDAgente	Nome	Zona
Bia	Bianchi	Mord
Bru	Bruni	Centro
Ner	Neri	Centro
Ros	Rossi	Sud
Ver	Verdi	Isole

Si osservi che nel realizzare una proiezione in cui non si includa una chiave per la relazione, i sistemi di database **reali** non effettuano alcun controllo sull'unicità delle righe della relazione risultante a meno che non sia esplicitamente richiesto (clausola DISTINCT, vedremo). Ne può risultare una tabella con record ripetuti. Questo problema non c'è nell'algebra relazionale che restituisce insiemi matematici, quindi per definizione di insieme non ci saranno mai tuple ripetute.

Calcoliamo: la **proiezione** su RagioneSoc e PartitaIVA della **selezione** di CLIENTE per Provincia = "BA":

$\Pi_{\text{RagioneSoc, PartitaIva}}(\sigma_{Provincia="BA"}(CLIENTE))$

RagioneSociale	PartitaIVA
Ottonifico BA	06357399181
Levigatoria …	01357739913

Calcoliamo l'equi-join delle tabelle Cliente ed Agente sulla condizione AGENTE.IdAgente = CLIENTE.IdAgente

AGENTE join CLIENTE on AGENTE.IDAgente = CLIENTE.IDAgente

IDCliente	RagioneSociale	PartitaIVA	Provincia	CLIENTE.IDAgente	AGENTE.IDAgente	Nome	Zona
Lami	Lamiere…	04357839912	TO	Bia	Bia	Bianchi	Mord
Levi	Levigatoria …	01357739913	FI	Ner	Ner	Neri	Centro
Luci	Lucidatura ..	02357639914	RM	Ner	Ner	Neri	Centro
Meta	Metallurgica	03357539916	BO	Ner	Ner	Neri	Centro
Metb	Metalli …	05357439917	NA	Ros	Ros	Rossi	Sud
Otto	Ottonifico BA	06357399181	BA	Ros	Ros	Rossi	Sud
Rame	Rame & Meta.	08357139932	PA	Ver	Ver	Verdi	Isole
Tond	Tondini …	09357039933	BS	Bia	Bia	Bianchi	Mord
Vite	Viteria MI…	01157939944	MI	Bia	Bia	Bianchi	Mord
Vitp	Viteria di …	01257839955	MI	Bia	Bia	Bianchi	Mord

In questo equi-join scompaiono le informazioni relative al cliente **'Prome'** Questo succede perché il codice agente di questo cliente è '*Gia*' e non alcun agente con questo identificativo nella tabella AGENTE.
Analogamente, L'agente con IDAgente = **'Bru'** non compare nel join perché non c'è alcun cliente al quale è stato assegnato. Si osservi che l'equi-join produce una relazione con informazioni ridondanti perché compre due volte il dato sull'agente. Per rimuovere una delle due colonne, si può operare una proiezione:
Se indichiamo con L il seguente insieme di attributi:
L = {IDCLiente, RagioneSociale, PartitaIVA, Provincia , CLIENTE.IDAgente, Nome, Zona }, allora sarà:

Π_L(AGENTE join CLIENTE on AGENTE.IDAgente = CLIENTE.IDAgente)

IDCliente	RagioneSoc	PartitaIVA	Provincia	CLIENTE.IDAgente	Nome	Zona
Lami	Lamiere…	04357839912	TO	Bia	Bianchi	Mord
Levi	Levigatoria …	01357739913	FI	Ner	Neri	Centro
Luci	Lucidatura ..	02357639914	RM	Ner	Neri	Centro
Meta	Metallurgica	03357539916	BO	Ner	Neri	Centro
Metb	Metalli …	05357439917	NA	Ros	Rossi	Sud
Otto	Ottonifico BA	06357399181	BA	Ros	Rossi	Sud
Rame	Rame & Meta.	08357139932	PA	Ver	Verdi	Isole
Tond	Tondini …	09357039933	BS	Bia	Bianchi	Mord
Vite	Viteria MI…	01157939944	MI	Bia	Bianchi	Mord
Vitp	Viteria di …	01257839955	MI	Bia	Bianchi	Mord

In cui è stata rimosso l'attributo AGENTE.IDAgente dall'equi-join.

Calcoliamo il Left outer join tra le tabelle CLIENTE ed AGENTE sulla condizione CLIENTE.IDAgente = AGENTE.IDAgente, cioè Calcoliamo:

AGENTE left join CLIENTE on AGENTE.IDAgente = CLIENTE.IDAgente

IDCliente	RagioneSociale	PartitaIVA	Provincia	CLIENTE.IDAgente	AGENTE.IDAgente	Nome	Zona
Lami	Lamiere…	04357839912	TO	Bia	Bia	Bianchi	Mord
Levi	Levigatoria …	01357739913	FI	Ner	Ner	Neri	Centro
Luci	Lucidatura ..	02357639914	RM	Ner	Ner	Neri	Centro
Meta	Metallurgica	03357539916	BO	Ner	Ner	Neri	Centro
Metb	Metalli …	05357439917	NA	Ros	Ros	Rossi	Sud
Otto	Ottonifico BA	06357399181	BA	Ros	Ros	Rossi	Sud
Rame	Rame & Meta.	08357139932	PA	Ver	Ver	Verdi	Isole
Tond	Tondini …	09357039933	BS	Bia	Bia	Bianchi	Mord
Vite	Viteria MI…	01157939944	MI	Bia	Bia	Bianchi	Mord
Vitp	Viteria di …	01257839955	MI	Bia	Bia	Bianchi	Mord
Prome	Prodotti…	07357239919	BG	Gia	**NULL**	NULL	NULL

Calcoliamo i clienti che non hanno un agente:

$\sigma_{AGENTE.IdAgente\ IS\ NULL}$(AGENTE left join CLIENTE on AGENTE.IDAgente = CLIENTE.IDAgente)

IDCliente	RagioneSociale	PartitaIVA	Provincia	CLIENTE.IDAgente	AGENTE.IDAgente	Nome	Zona
Prome	Prodotti…	07357239919	BG	Gia	**NULL**	NULL	NULL

Calcoliamo: AGENTE right join CLIENTE on AGENTE.IDAgente = CLIENTE.IDAgente

IDCliente	RagioneSociale	PartitaIVA	Provincia	CLIENTE.IDAgente	AGENTE.IDAgente	Nome	Zona
Lami	Lamiere…	04357839912	TO	Bia	Bia	Bianchi	Mord
Levi	Levigatoria …	01357739913	FI	Ner	Ner	Neri	Centro
Luci	Lucidatura ..	02357639914	RM	Ner	Ner	Neri	Centro
Meta	Metallurgica	03357539916	BO	Ner	Ner	Neri	Centro
Metb	Metalli …	05357439917	NA	Ros	Ros	Rossi	Sud
Otto	Ottonifico BA	06357399181	BA	Ros	Ros	Rossi	Sud
Rame	Rame & Meta.	08357139932	PA	Ver	Ver	Verdi	Isole
Tond	Tondini …	09357039933	BS	Bia	Bia	Bianchi	Mord
Vite	Viteria MI…	01157939944	MI	Bia	Bia	Bianchi	Mord
Vitp	Viteria di …	01257839955	MI	Bia	Bia	Bianchi	Mord
NULL	**NULL**	**NULL**	**NULL**	**NULL**	Bru	Bruni	Centro

Si osservi che nel right join è stata inclusa la tupla <Bru,Bruni, Centro> della relazione AGENTE che non ha una controparte nella relazione CLIENTE.

Calcoliamo ora: AGENTE full join CLIENTE on AGENTE.IDAgente = CLIENTE.IDAgente

IDCliente	RagioneSociale	PartitaIVA	Provincia	CLIENTE.IDAgente	AGENTE.IDAgente	Nome	Zona
Lami	Lamiere…	04357839912	TO	Bia	Bia	Bianchi	Mord
Levi	Levigatoria …	01357739913	FI	Ner	Ner	Neri	Centro
Luci	Lucidatura ..	02357639914	RM	Ner	Ner	Neri	Centro
Meta	Metallurgica	03357539916	BO	Ner	Ner	Neri	Centro
Metb	Metalli …	05357439917	NA	Ros	Ros	Rossi	Sud
Otto	Ottonifico BA	06357399181	BA	Ros	Ros	Rossi	Sud
Rame	Rame & Meta.	08357139932	PA	Ver	Ver	Verdi	Isole
Tond	Tondini …	09357039933	BS	Bia	Bia	Bianchi	Mord
Vite	Viteria MI…	01157939944	MI	Bia	Bia	Bianchi	Mord
Vitp	Viteria di …	01257839955	MI	Bia	Bia	Bianchi	Mord
Prome	**Prodotti…**	**07357239919**	**BG**	**Gia**	**NULL**	NULL	NULL
NULL	**NULL**	**NULL**	**NULL**	**NULL**	Bru	Bruni	Centro

Il join risultante include la concatenzazione delle tuple di AGENTE e CLIENTE che verificano la condizione di join più tutte le tuple sia di AGENTE che di CLIENTE che non hanno una controparte nell'altra tabella. Le tuple senza controparti corrisponndenti sono completate con VALORI NULL.

L'importanza dell'outer join risiede nel fatto che ci permette di risolvere i problemi di assenza, cioè problemi nei quali bisogna saper rispondere a quesiti del tipo: Quale agente non ha clienti? Quale cliente non ha un agente assegnato? Chi sono i clienti che non hanno fatto acquisti? Chi sono gli studenti senza voti?

Esercizi

ESESRCIZIO 1

Considera Il seguente schema di base dati:
STUDENTE(Matricola, Cognome, Nome, CodScuola)
SCUOLA(CodScuola, NomeScuola, Citta)
Scrivi in algebra relazionale il calcolo per estrarre dalla base dato il nome e cognome degli studenti che frequentano l'istituto Carli, che ha codice: "SIA_THE_BEST"

ESERCIZIO 2

Data la seguente base dati:

IMPIEGATO			
Matricola	Cognome	Nome	Data di nascita
12	Rossi	Tino	1/1/1970
13	Neri	Tano	1/1/1960
14	Bianchi	Tono	1/1/1970
15	Verdi	Tuno	1/1/1980

PARTECIPAZIONE	
Impiegato	Progetto
12	1
13	1
13	2
14	5

PROGETTO		
ID	NomeProgetto	Costo
1	Astro	400
5	Cosmo	800
2	Stella	100

Calcola:

- $\sigma_{Data\ di\ nascita > 1/1/1970' \ AND \ Matricola < 15}$ *(IMPIEGATO)*
- $\Pi_{Cognome,\ Nome}$(IMPIEGATO)
- $\Pi_{Cognome,\ Nome}(\sigma_{Data\ di\ nascita > 1/1/1970' \ AND \ Matricola < 15}$ *(IMPIEGATO))*
- IMPIEGATO x PARTECIPAZIONE
- IMPIEGATO join PARTECIPAZIONE on Matricola=Impiegato
- (IMPIEGATO left join PARTECIPAZIONE on Matricola = Impiegato) join PROGETTO on Progetto= ID
- PROGETTO join PARTECIPAZIONE on ID=Impiegato
- $\Pi_{NomeProgetto,\ Impiegato}$(PROGETTO join PARTECIPAZIONE on ID=Impiegato)
- $\Pi_{NomeProgetto,\ Impiegato}(\ \sigma_{Costo > 400}$ (PROGETTO join PARTECIPAZIONE on ID=Impiegato*))*
- IMPIEGATO left join (PROGETTO join PARTECIPAZIONE on ID=Impiegato) on Matricola = Impiegato
- IMPIEGATO right join (PROGETTO join PARTECIPAZIONE on ID=Impiegato) on Matricola = Impiegato
- IMPIEGATO full join (PROGETTO join PARTECIPAZIONE on ID=Impiegato) on Matricola = Impiegato

ESERCIZIO 3

Data la seguente base dati:

VOLO			
Numero	Rotta	Data	Comandante
1	AZ1	23/11/2019	Wing
2	AZ1	24/12/2019	Ming
3	CXF	02/01/2020	Ting

ROTTA		
Codice	Partenza	Arrivo
AZ1	FCO	JFK
CXF	LAX	FCO
XXX	SCH	UNI

PRENOTAZIONE		
Volo	Classe	Cliente
1	Economy	Parsley
1	Business	Sage
3	Economy	Rosmary

Calcola:

- $\Pi_{\text{Numero, Rotta}}$ (VOLO) **X** ROTTA
- $\Pi_{\text{Numero, Rotta}}$ (VOLO) $\bowtie_{\text{Rotta = Codice}}$ ROTTA
- VOLO left join PRENOTAZIONE on Numero = Volo
- VOLO left join ROTTA on Rotta = Codice
- VOLO right join ROTTA on Rotta = Codice
- VOLO full join ROTTA on Rotta = Codice

ESERCIZIO 3

Una organizzazione internazionale raggruppa gli iscritti di diverse nazioni.

- Gli iscritti possono aderire con ruoli diversi: socio ordinario, sostenitore, affiliato;
- l'importo delle quote di iscrizione + libero e viene versato con pagamenti che possono essere fatti in date diverse (anche più versamenti nello stesso anno senza controlli di scadena).

Dalla descrizione fornita si capisce che devono essere descritti nel sistema

- gli iscritti, per rappresentare le informazioni riguardanti i soci dell' organizzazione
- le nazioni, per gli stati di appartenenza degli iscritti, essendo l'organizzazione di tipo internazionale.
- Il tipo di adesione, per i diversi tipi di ruolo con i quali i soci aderiscono;
- i pagamenti, per i versamenti delle quote di iscrizione.

Considera allora il seguente schema logico dei dati:
TIPO_ADESIONE(CodTipo, Descrizione)
NAZIONE(CodNazione, Descrizione)
ISCRITTO(CodIscritto, Cognome, Nome, Telefono, CodiceTipoAdesione, CodiceNazione)
PAGAMENTO(ID, Data, Importo, CodiceIscritto)

In termini di gestione dei dati, Le tabelle TIPO_ADESIONE e NAZIONE hanno la funzione di un dizionario di dati e nel tempo subiranno poche modifiche. Queste tabelle vengono popolate all'atto della creazione del database. Le tabelle ISCRITTO e PAGAMENTO invece saranno gestite da una applicazione che metterà a disposizione le funzionalità per inserire, cancellare, modificare i record delle relazioni.

Partendo dallo schema logico del database:

- Quali potrebbero essere le chiavi primarie?
- Vanno imposto vincoli di integrità referenziale?
- Scrivi in algebra relazionale le seguenti interrogazioni:
    - Elenco di tutti gli iscritti con cognome, nome, telefono
    - L'elenco degli iscritti con cognome, nome, tipo socio, per la nazione 'x'
    - Tutti i dati degli iscritti di tipo 'sostenitore'
    - L'elenco dei pagamenti con cognome, data, importo effettuata nell'anno in corso con cognome e nome degli iscritti che hanno effettuato i pagamenti (data compresa tra 1/01/2019 e 31/12/2019)
    - L'elenco dei pagamenti effettuati dall'iscritto con nome e cognome 'myName', 'mySur'
    - L'elenco degli iscritti che non hanno effettuato pagamenti