Algebra Relazionale

Lucia Ferrari lucia02.ferrari@edu.unife.it

Introduzione

Introduzione

L'insieme principale delle operazioni per il modello relazionale è l'algebra relazionale.

Il risultato di un'espressione è una nuova relazione che può essere usata all'interno di un'altra espressione.

Operazioni relazionali unarie:

SELECT, PROJECT e RENAME

Operazione: SELECT (selezione)

L'operazione di selezione è utilizzata per scegliere un sottoinsieme di tuple (un insieme di RIGHE) della relazione che soddisfano una certa condizione. Può essere vista come una partizione ORIZZONTALE della relazione in due insiemi: le tuple che soddisfano la condizioni e quelle che non lo fanno.

La sintassi è:

$$\sigma_{\langle \text{condizione di selezione} \rangle}(R),$$

dove σ (sigma) denota l'operatore di selezione e la condizione di selezione è un'espressione booleana (ha valore vero o falso) specificata sugli attributi della relazione R.

L'espressione booleana all'interno di $\langle\rangle$ è formata da un certo numero di clausole che possono essere in due forme:

$$\langle\text{attribute name}\rangle \text{ operazione } \langle\text{constant value}\rangle \text{ oppure}$$
$$\langle\text{attribute name}\rangle \text{ operazione } \langle\text{attribute name}\rangle$$

dove:

- $\langle$attribute name$\rangle$ è un attributo di R
- operazione $\in \{<, \leq, =, \neq, \geq, >\}$
- $\langle$constant value$\rangle$ è una costante che appartiene al dominio dell'attributo

Le clausole possono essere connesse tra loro tramite: and ($\wedge$), or ($\vee$), o not ($\neg$).

Ad esempio, considerando la seguente relazione R:

Nome	Cognome	Salario	Eta
Maria	*Rossi*	1500	22
Giovanni	*Giovanni*	900	20
Anna	*Bianchi*	5000	30
Maria	*Maria*	1200	45

1) Dopo l'operazione : $\sigma_{(Salario>3000)}(R)$ la relazione risultante è:

Nome	Cognome	Salario	Eta
Anna	*Bianchi*	5000	30

2) Dopo l'operazione : $\sigma_{(Nome=Cognome)\wedge(Salario>500)}(R)$ la relazione risultante è:

Nome	Cognome	Salario	Eta
Giovanni	*Giovanni*	900	20
Maria	*Maria*	1200	45

SELECT Operation$_{(4)}$

L'operatore di selezione è commuttativo:

$$\sigma_{\langle \text{cond}_1 \rangle}(\sigma_{\langle \text{cond}_2 \rangle}(R)) = \sigma_{\langle \text{cond}_2 \rangle}(\sigma_{\langle \text{cond}_1 \rangle}(R)).$$

Esempio:

$$\sigma_{\langle \text{Nome} = \text{"Maria"} \rangle}(\sigma_{\langle \text{Eta} > 30 \rangle}(R)) = \sigma_{\langle \text{Eta} > 30 \rangle}(\sigma_{\langle \text{Nome} = \text{"Maria} \rangle}(R)).$$

Ovvero:

Nome	Cognome	Salario	Eta
Maria	*Maria*	1200	45

Operazione: PROJECT (proiezione)

L'operatore di proiezione seleziona alcune COLONNE dalla tabella e ne elimina delle altre.
Può essere vista come una partizione verticale: seleziona solo alcuni attributi della relazione.

La sintassi è:

$$\pi_{\langle \text{lista di attributi} \rangle}(R)$$

dove π (pi) è il simbolo usato per rappresentare l'operatore `PROJECT` e $\langle$lista di attributi$\rangle$ è la lista di attributi (separati da una virgola) che si vuole ottenere dalla relazione R.

Il risultato dell'operazione di proiezione contiene solo gli attributi specificati nella ⟨lista attributi⟩ e nello stesso ordine in cui appaiono nella lista.
Inoltre l'operazione di proiezione RIMUOVE eventuali tuple duplicate (in modo che la relazione rimanga valida)

Ad esempio considerando:

Id	Articolo	Prezzo	Scaffale
1	*Torcia*	100	1
2	*Bilancia*	90	2
3	*Martello*	70	5
4	*Martello*	100	5

Dopo $\pi_{Id,Articolo}(R)$ la relazione risultante è:

Id	Articolo
1	*Torcia*
2	*Bilancia*
3	*Martello*
4	*Martello*

Dopo $\pi_{Articolo,Scaffale}(R)$ la relazione risultante è:

Articolo	Scaffale
Torcia	1
Bilancia	2
Martello	5

Sequenza di operazioni

In generale per la maggiorparte di query abbiamo bisogno di applicare più operazioni di diverso tipo una dopo l'altra. Possiamo quindi:

- Scrivere l'operazione come una singola espressione innestando i risultati
- Applicare un'operazione alla volta salvando i risultati uno alla volta.

Per esempio la singola espressione:

$$\pi_{FName,LName,Salary}(\sigma_{DNo=5}(EMPLOYEE)),$$

può essere espressa da:

$$\begin{aligned} DEPT5\_EMPS &\leftarrow \sigma_{DNo=5}(EMPLOYEE) \\ RESULT &\leftarrow \pi_{FName,LName,Salary}(DEPT5\_EMPS), \end{aligned}$$

dove $\leftarrow$ è l'operatore di assegnazione

Operatore: RENAME

Se non è applicata nessuna operazione di rinomina il nome degli attributi nella relazione risultante è lo stesso di quelli della relazione di partenza e anche l'ordine (a meno che non venga modificato da un'operazione di proiezione). L'operazione RENAME applicata alla relazione R con grado n ha la seguente sintassi:

$$\rho_{S(B_1,\ldots,B_n)}(R) \quad \text{o} \quad \rho_S(R) \quad \text{o} \quad \rho_{(B_1,\ldots,B_n)}(R),$$

dove ρ (rho) è il simbolo usato per l'operatore `RENAME`, S è il nuovo nome della relazione, e $B_1, \ldots, B_n$ sono i nuovi nomi dei suoi attributi.

- La prima espressione rinomina sia la relazione che i suoi attributi
- La seconda solo la relazione
- La terza rinomina solo gli attributi.

Considerando la relazione R:

Nome	Cognome
Mario	*Rossi*
Elena	*Bianchi*

Dopo $\rho_{S(NomeImp, CognomeImp)}(R)$, otteniamo la relazione S:

NomeImp	CognomeImp
Mario	*Rossi*
Elena	*Bianchi*

Algebra relazionale: operazioni insiemistiche

Compatibilità

Le operazioni insiemistiche (UNIONE, INTERSEZIONE, DIFFERENZA) devono risultati compatibili all'unione, ovvero due relazioni $R(A_1, \ldots, A_n)$ e $S(B_1, \ldots, B_n)$ sono dette compatibili se:

- Hanno lo stesso grado n (stesso numero di attributi)
- $dom(A_i) = dom(B_i)$, $1 \leq i \leq n$.

Nella relazione finale per convenzione si tengono i nomi degli attributi di R (ovvero della prima)

Operazioni: Unione, Intersezione, Differenza

- `UNION`: Il risultato dell'operazione, denotato da $R \cup S$, è una relazione che include tutte le tuple che sono: in R o in S oppure in entrambi. Tuple duplicate vengono eliminate.
- `INTERSECTION`: Il risultato, denotato da $R \cap S$, è una relazione che include tutte le tuple che sono sia in R che in S.
- `SET DIFFERENCE`: Il risultato dell'operazione, denotato da $R - S$, è una relazione che include tutte le tuple che sono in R ma non in S.

Considerando le seguenti relazioni R e S:

A	B
α	10
α	20
β	30

R

C	D
α	20
β	40
β	50

S

Risultati:

A	B
α	10
α	20
β	30
β	40
β	50

$R \cup S$

A	B
α	20

$R \cap S$

A	B
α	10
β	30

$R - S$

Proprietà delle operazioni insiemistiche

- Unione e intersezione sono operazioni commutative:

$$R \cup S = S \cup R \quad \text{and} \quad R \cap S = S \cap R,$$

- Unione e intersezione sono anche associative:

$$R \cup (S \cup T) = (R \cup S) \cup T \quad \text{and} \quad R \cap (S \cap T) = (R \cap S) \cap T.$$

- La differenza NON è commutativa

$$R - S \neq S - R.$$

- L'intersezione può essere espressa tramite unione e differenza:

$$R \cap S = ((R \cup S) - (R - S)) - (S - R).$$

Prodotto Cartesiano

Il prodotto cartesiano, denotato da $\times$, è un'altra operazione insiemistica ma non ha bisogno di avere relazioni compatibili.

Il risultato di $R(A_1, \ldots, A_n) \times S(B_1, \ldots, B_m)$ è una relazione

$$Q(A_1, \ldots, A_n, B_1, \ldots, B_m)$$

di grado $n + m$ con gli attributi nello stesso ordine.
Se R ha n_R tuple e S n_S tuple allora $R \times S$ avrà $n_R \cdot n_S$ tuple.

Considerando le relazioni R e S:

Nome	Progetti
anna	10
barbara	20
claudia	30

R

Codice	Dip
1	Math
2	CS

S

Allora:

Nome	Progetti	Codice	Dip
anna	10	1	Math
anna	10	2	CS
barbara	20	1	Math
barbara	20	2	CS
claudia	30	1	Math
claudia	30	2	CS

$R \times S$

Operazioni relazionali binarie: JOIN e DIVISION

Operazione: JOIN

L'operazione di JOIN effettua un prodotto cartesiano tra due relazioni e poi ci applica una selezione

La sintassi dell'operazione di JOIN tra due relazioni $R(A_1, \ldots, A_n)$ e $S(B_1, \ldots, B_m)$ è:

$$R \bowtie_{\langle \text{join condition} \rangle} S.$$

Il risultato dell'operazione di JOIN è una relazione Q con $n + m$ attributi

$$Q(A_1, \ldots, A_n, B_1, \ldots, B_m)$$

Q ha una tupla per ogni combinazione di tuple di R e S che soddisfano la condizione specificata.

Una condizione di join generale è nella forma:

$$\langle \text{cond} \rangle \wedge \langle \text{cond} \rangle \wedge \ldots \wedge \langle \text{cond} \rangle,$$

dove ogni $\langle \text{cond} \rangle$ è nella forma $A_i \theta B_j$, A_i è un attributo di R, B_j è un attributo di S, A_i e B_i hanno lo stesso dominio, e $\theta \in \{<, \leq, =, \neq, \geq, >\}$.

Una JOIN con queste condizioni è chiamata THETA-JOIN.
Le tuple i cui attributi sono NULL o per cui le condizione di JOIN risulta falsa non appaiono nel risultato.

Quando due condizioni di JOIN hanno unicamente $=$ come operatore sono chiamate EQUI-JOIN.
In questi casi nelle tuple risultanti appaiono sempre due o più coppie di attributi che hanno lo stesso identico valore

Per evitare la presenza di questi valori doppi si può usare una operazione di NATURAL JOIN, indicata da $*$. Per poterla utilizzare le due relazioni devono avere degli attributi con lo stesso nome (o è necessaria una operazione di rinomina).

Considerando le due relazioni *Impiegato* e *Dipartimento*:

Nome	Iniziali	Id_dip
Anna	*AF*	1
Barbara	*BG*	1
Claudio	*CO*	3
Davide	*DL*	2
Emma	*EE*	2

Impiegato

Cod_dip	Indirizzo
1	*ViaM*30
2	*ViaP*10

Dipartimento

Utilizzando EQUI-JOIN diventa:

Nome	Iniziali	Id_dip	Cod_dip	Indirizzo
Anna	*AF*	1	1	*ViaM*30
Barbara	*BG*	1	1	*ViaM*30
Davide	*DL*	2	2	*ViaP*10
Emma	*EE*	2	2	*ViaP*10

$$Impiegato \bowtie_{(Impiegato.Id\_dip = Diparimento.Cod\_dip)} Dipartimento$$

Utilizzando JOIN NATURALE diventa:

Nome	Iniziali	Id_dip	Indirizzo
Anna	*AF*	1	*ViaM*30
Barbara	*BG*	1	*ViaM*30
Davide	*DL*	2	*ViaP*10
Emma	*EE*	2	*ViaP*10

$$\rho_{(Id\_dip, Indirizzo)}(Dipartimento)$$

$$Impiegato *_{(Impiegato.Id\_dip = Diparimento.Id\_dip)} Dipartimento$$

Insieme completo di operazioni dell'algebra

Può essere mostrato che l'insieme $\{\sigma, \pi, \rho, \cup, -, \times\}$ è un insieme completo. Ovvero ogni altra operazione dell'algebra relazionale può essere ottenuta con una combinazione di operazioni di questo insieme.

Per esempio:

$$\begin{aligned} R \cap S &\equiv (R \cup S) - ((R - S) \cup (S - R)) \\ R \bowtie_{\langle \text{condition} \rangle} S &\equiv \sigma_{\langle \text{condition} \rangle}(R \times S). \end{aligned}$$

Operazione: Division

L'operazione di divisione, denotata da $\div$, è applicata a due relazioni $R(Z)$ e $S(X)$, dove gli attributi di S sono un sottoinsieme di quelli di R; ovvero $X \subseteq Z$.
Prendiamo $Y = Z - X$.

Il risultato di $R(Z) \div S(X)$ è una relazione $T(Y)$ che include una tupla t della relazione R solo se t[Y] appare combinato con ognuna delle tuple di S

Considerando le due relazioni R e S:

A	B
α	1
β	1
α	3
α	2
β	2
β	4

R

A
α
β

S

Diventa:

B
1
2

$R \div S$

Per ottenere il risultato della divisione:

1. prendiamo gli attributi che non sono presenti in S e che appaiono in R
2. Prendiamo solo le tuple che hanno quel attributi combinato con TUTTI i valori di S.

Riassunto

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$ OR $R_1 * R_2$
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Altre operazioni relazionali

Funzioni di Aggregazione

Una funzione di aggregazione $\mathcal{F}$, è definita nel seguente modo:

$$\langle \text{grouping attributes} \rangle \mathcal{F}_{\langle \text{function list} \rangle}(R),$$

dove:
- ⟨grouping attributes⟩ è una lista di attributi della relazione R.
- ⟨function list⟩ è una lista di coppie (⟨function⟩⟨attribute⟩).

In ognuna delle coppie:
-> ⟨function⟩ è una delle funzioni consentite tra: `SUM`, `AVERAGE`, `MAXIMUM`, `MINIMUM`, `COUNT`
-> ⟨attribute⟩ è un attributo della relazione R.

Il risultato finale avrà gli attributi di raggruppamento + i risultati delle funzioni applicate

Considerando R:

SSN	Name	Salary	Dno
1	A	30000	5
2	B	40000	5
3	C	25000	4
4	D	30000	4
5	E	20000	4
6	F	35000	1

1
2
3

Allora:

Dno	N_Emp	Avg_Sal
5	2	35000
4	3	25000
1	1	35000

1°
2°
3°

Count(SSN)	Average(Salary)
6	30000

2) $\mathcal{F}_{COUNT(SSN),AVERAGE(Salary)}(R)$

1) $\rho_{(Dno,N\_Emp,Avg\_Sal)}({}_{Dno}\mathcal{F}_{COUNT(SSN),AVERAGE(Salary)}(R))$

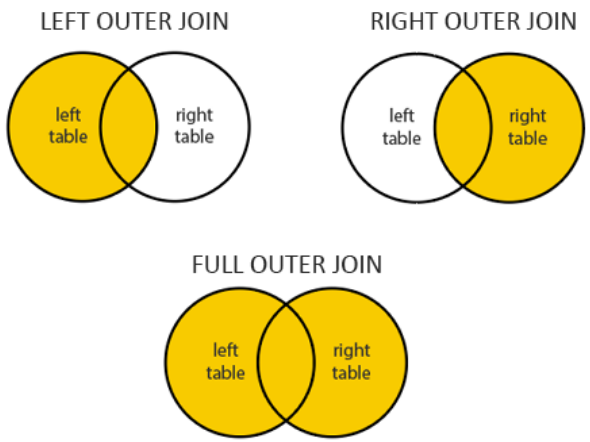
1) Calcola la media dei salari considerando i GRUPPI di tuple che hanno lo stesso Dno e conta quante tuple hanno lo stesso valore di Dno

2) Conta tutte le tuple presenti nella relazione R e fa una media del salario!

Operazione: OUTER-JOIN

L'operazione di OUTER-JOIN è utilizzata per effettuare una join che non escluda le tuple che non hanno corrispondenza nell'altra relazione o che hanno valore NULL sulla condizione di join.
Tipologie:

- LEFT-OUTER-JOIN tra R e S, denotata da $R \ltimes_{\langle \text{cond} \rangle} S$, mantiente tutte le tuple di R (della tabella di sinistra).
- RIGHT-OUTER-JOIN tra R e S, denotata da $R \rtimes_{\langle \text{cond} \rangle} S$, tiene tutte le tuple di S (della tabella di destra).
- FULL-OUTER-JOIN tra R e S, denotata da $R \ltimes\!\rtimes_{\langle \text{cond} \rangle} S$, tiene tutte le tuple sia di R che di S (entrambe le tabelle).



Considerando la seguente relazione *Imp* e *Dip*:

Nome	Cod
Andrea	1
Barbara	2
Claudia	4

Imp

Cod	Ind
1	*ViaA*1
2	*ViaB*2
3	*ViaC*3

Dip

Allora $Imp \bowtie_{Imp.Cod=Dip.Cod} Dip$ rispettivamente con LEFT-JOIN, RIGHT-JOIN E FULL-JOIN:

Nome	Cod	Ind
Andrea	1	*ViaA*1
Barbara	2	*ViaB*2
Claudia	4	NULL

Nome	Cod	Ind
Andrea	1	*ViaA*1
Barbara	2	*ViaB*2
NULL	3	*ViaC*3

Nome	Cod	Ind
Andrea	1	*ViaA*1
Barbara	2	*ViaB*2
NULL	3	*ViaC*3
Claudia	4	NULL

Esercizi

Esercizio

Sia dato il seguente schema relazionale:

$$Person(\underline{name}, age, gender)$$
$$Frequents(\underline{name}, \underline{pizzeria})$$
$$Eats(\underline{name}, \underline{pizza})$$
$$Serves(\underline{pizzeria}, pizza, price).$$

Esprimere in algebra relazionale le seguenti interrogazioni senza usare l'operatore di divisione $\div$).

Person = informazioni sui clienti (nome, età, sesso)
Frequents = elenco delle pizzerie che ogni persona frequenta
Eats = elenco delle pizze che ogni persona mangia (= quelle che gli piacciono)
Serves = elenco delle pizzerie, le pizze che offrono e il loro prezzo

Person(<u>name</u>, age, gender)
Frequents(<u>name</u>, <u>pizzeria</u>)
Eats(<u>name</u>, <u>pizza</u>)
Serves(<u>pizzeria</u>, pizza, price).

1. Determinare i nomi di donne che mangiano la pizza con i funghi o la margherita o entrambe.

Mangiano ← Person * Eats
Pizze ← σ((pizza='funghi' ∨ pizza='margherita') ∧ gender='F')(Mangiano)
risultato ← π name (pizze)

Mangiano = associo le informazioni delle persone con le pizze che mangiano
Pizze = recupero le informazioni richieste dall'espressione
risultato = prendo solo i nomi delle persone

2. Determinare i nomi di donne che mangiano sia pizza margherita che funghi.

Mangiano ← Person * Eats
Funghi ← σ(pizza='funghi' ∧ gender='F') Mangiano
Margherita ← σ(pizza='margherita' ∧ gender='F') Mangiano
Result ← π name (Funghi) ∩ π name (Margherita)

Prendo le persone che mangiano pizze con i funghi, le persone che mangiano pizze magherite e faccio l'intersezione tra i due insiemi

Person(name, age, gender)
Frequents(name, pizzeria)
Eats(name, pizza)
Serves(pizzeria, pizza, price).

3. Tutte le pizzerie che servono almeno una pizza che Amy mangia e che costa meno di 10 euro.

PizzeMio ← Eats * Serves

Amy ← σ(name = "Amy" ∧ price < 10) Pizzerie

risultato ← π pizzeria Amy

Associo le informazioni sulle pizze mangiate con le pizzerie che le offrono, poi effettuo le operazioni di selezione e proiezione

4. Nomi di pizzerie frequentate da almeno una persona minorenne

Persone ← Frequents * Person

Risultato ← π pizzeria (σ age < 18) Persone

Associo le informazioni sulle persone con le pizzerie che frequentano e poi faccio le operazioni richieste

Person(<u>name</u>, age, gender)
Frequents(<u>name</u>, <u>pizzeria</u>)
Eats(<u>name</u>, <u>pizza</u>)
Serves(<u>pizzeria</u>, pizza, price).

5. Tutte le pizzerie frequentate solo da donne o solo da uomini

tutti ← Person * Frequents
Solo_F ← $\pi_{pizzeria}$(tutti) − $\pi_{pizzeria}(\sigma_{gender='M'}$ tutti
Solo_M ← $\pi_{pizzeria}$(tutti) − $\pi_{pizzeria}(\sigma_{gender='F'}$ tutti
risultato ← Solo_F ∪ Solo_M

Tutti = associo le informazioni delle persone con le pizzerie che frequentano
Solo_F = per trovare le pizzerie frequentate SOLO da donne all'insieme di tutte le pizzerie frequentate rimuovo quelle che sono frequentate anche da uomini.
Solo_M = " da donne
risultato = unisco i risultati ottenuti

6. Nomi di tutte le pizzerie che Dan non ha mai frequentato e in cui può comprare pizze che mangia.

Pizzerie_in_cui_mangia ← $\pi_{pizzerie}(\sigma_{name='Dan'}$(Eats * Serves))

R ← Pizzerie_in_cui_mangia − $\pi_{pizzeria}(\sigma_{name='Dan'}$ Frequents

Pizzerie in cui mangia = sono le pizzerie in cui sono servite delle pizze che piacciono a Dan (le potrebbe mangiare)
R = all'insieme delle pizzerie in cui potrebbe mangiare rimuovo quelle che ha frequentato

Person(name, age, gender)
Frequents(name, pizzeria)
Eats(name, pizza)
Serves(pizzeria, pizza, price).

7. Individuare per ogni persona le pizze che mangia e che NON sono servite dalle pizzerie che frequenta. (risultati nella forma: nome persona - pizza)

Pizze_frequenta ← $\pi_{name, pizza}$ (Frequents * Serves)

result ← Eats − Pizze_frequenta

Pizze_frequenta = estraggo nome e pizza dalle pizzerie che ogni persona frequenta
result = dalle pizze che ogni persona potrebbe mangiare rimuovo le pizze che già mangiano nelle pizzerie che frequentano

8. Nomi di tutte le persone che frequentano SOLO pizzerie che servono almeno una pizza che mangiano

Non_mangiano ← π_{name} (Frequents − $\pi_{name, pizzeria}$ (Eats * Serves))

Result ← π_{name} (Person) − Non_mangiano

Non_mangiano = ottengo i nomi di persone che frequentano pizzerie che servono anche pizze che non mangiano (non gliene piace neanche una)
Result = tolgo da tutte le persone quelle che frequentano le pizzerie in cui non mangiano nulla (Non_mangiano) ottenendo solo le persone che frequentano pizzerie in cui possono mangiare qualcosa

Person(name, age, gender)
Frequents(name, pizzeria)
Eats(name, pizza)
Serves(pizzeria, pizza, price).

9. Nomi delle persone che frequentano TUTTE le pizzerie che servono almeno una pizza che mangiano

mangiano ← $\Pi_{name, pizzeria}$ (Eats ⋈ Serves)

Non_frequentano ← Π_{name} (Mangiano - Frequents)

Risultato ← Π_{name} (Person) - Non_frequentano

mangiano = nomi di persona e nomi delle pizzerie in cui queste persone mangiano qualcosa
Non_frequentano = ottengo i nomi delle persone che non frequentano TUTTE le pizzerie in cui possono mangiare (esempio: se a tizio piacciono le pizze servite da A, B, C ma frequenta solo A,B significa non frequenta tutte le pizzerie che offrono le pizze che gli piacciono, infatti A,B,C - A,B = C -> C pizzeria che non frequenta)
risultato = da tutte le persone rimuovo il risultato di non_frequentano

10. Ricavare il prezzo medio delle pizze

R ← $\mathcal{G}_{AVERAGE\ (PRICE)}$ Serves

Person(name, age, gender)
Frequents(name, pizzeria)
Eats(name, pizza)
Serves(pizzeria, pizza, price).

11. Ricavare il prezzo medio delle pizze per ognuna delle pizzerie

$R \leftarrow {}_{\text{pizzeria}}\mathcal{G}_{\text{AVERAGE(PRICE)}}\ \text{Serves}$

12. Ricavare il prezzo medio di ogni pizza che mangia Amy (nome pizza - prezzo medio)

$\text{Amy} \leftarrow \sigma_{\text{name}='\text{Amy}'}(\text{Eats} * \text{Serves})$

$R \leftarrow {}_{\text{pizza}}\mathcal{G}_{\text{AVERAGE(PRICE)}}\ \text{Amy}$

Person(name, age, gender)
Frequents(name, pizzeria)
Eats(name, pizza)
Serves(pizzeria, pizza, price).

13. Ricavare il numero di persone che mangia ogni pizza (numero persone - pizza)

$R \leftarrow pizza\, \mathcal{G}_{COUNT}(NAME)\ Eats$

14. Ricavare il prezzo della pizza con costo massimo e il prezzo della pizza con costo minimo

$MAX \leftarrow \mathcal{G}_{MAX(PRICE)}\ Serves$

$MIN \leftarrow \mathcal{G}_{MIN(PRICE)}\ Serves$

Person(name, age, gender)
Frequents(name, pizzeria)
Eats(name, pizza)
Serves(pizzeria, pizza, price).

15. Ricavare per ogni pizzeria la pizza che costa di più e il suo prezzo (pizzeria - pizza - prezzo)

$$S2 \leftarrow \rho_{S_2}(pizz2, p2, price2)(Serves)$$

$$No\_max \leftarrow Serves \bowtie (pizzeria = pizz2 \wedge price < price2)\ S_2$$

$$Risultato \leftarrow Serves - No\_max$$

S2 = è una copia di Serves
No_max = ottiene le informazioni delle pizze che non hanno prezzo massimo
risultato = da tutte le pizze rimuove quelle che non hanno prezzo massimo ottenendo SOLO quelle di prezzo massimo