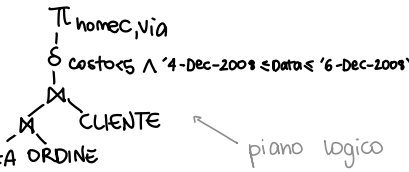
Tipologia A

SELECT NomeC, Via
FROM PIZZA NATURAL JOIN ORDINE NATURAL JOIN CLIENTE
WHERE Costo < 5 AND Data BETWEEN (DATE '4-Dec-2008' AND DATE '6-Dec-2008');

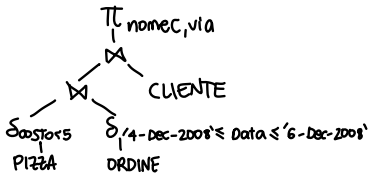
$\pi_{nomeC,via}(\delta_{costo<5 \wedge '4\text{-}dec\text{-}2008' \le data \le '6\text{-}dec\text{-}2008'}(PIZZA \bowtie ORDINE \bowtie CLIENTE))$

espressione algebrica canonica

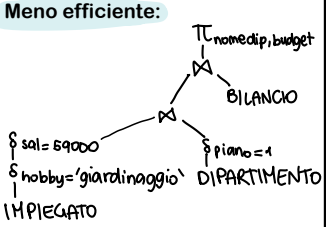


piano logico

Ci sono due modi per renderlo più efficiente: anticipando o la selezione o la proiezione



Meno efficiente:



SELECT NomeC, Via
FROM PIZZA NATURAL JOIN ORDINE NATURAL JOIN CLIENTE
WHERE Costo < 5 AND Data BETWEEN (DATE '4-Dec-2008' AND DATE '6-Dec-2008');

Fattori booleani per la query

4. Cosa cambierebbe relativamente ai fattori booleani se la query fosse:

SELECT NomeC, Via
FROM PIZZA NATURAL JOIN ORDINE NATURAL JOIN CLIENTE
WHERE Costo < 5 OR Data BETWEEN (DATE '4-Dec-2008' AND DATE '6-Dec-2008');

Se c'è OR, non esistono fattori booleani per la query

SELECT nomedip, budget
FROM IMPIEGATO NATURAL JOIN DIPARTIMENTO NATURAL JOIN BILANCIO
WHERE piano = 1 AND sal = 59000 AND hobby = "giardinaggio";

e le seguenti informazioni sullo schema fisico:

- indici ordinati (B+ alberi) su IMPIEGATO.eid, DIPARTIMENTO.piano, DIPARTIMENTO.did e BILANCIO.did; indice hash su IMPIEGATO.sal.
- i salari distinti degli impiegati sono 500, gli hobby diversi degli impiegati sono 200 e la compagnia occupa due piani dell'edificio;
- nella base di dati ci sono 50.000 impiegati e 5.000 dipartimenti (ognuno con le relative informazioni sul bilancio);

4. Supponendo che nel piano logico di esecuzione tutte le selezioni vengano eseguite prima del join mentre la proiezione venga eseguita dopo, stimare quante tuple verranno selezionate per la relazione IMPIEGATO.

La formula è F(C1 AND C2) = F(C1)*F(C2), quindi: F(sal=59000)*F(hobby='giardinaggio') Usiamo quindi la formula F(A=v) = 1/V(A,R) Quindi 1/500*1/200 = 1/100000, con 500=numero salari, 200=numero hobby, 50000= numero dipendenti 50000*1/100000=1/2

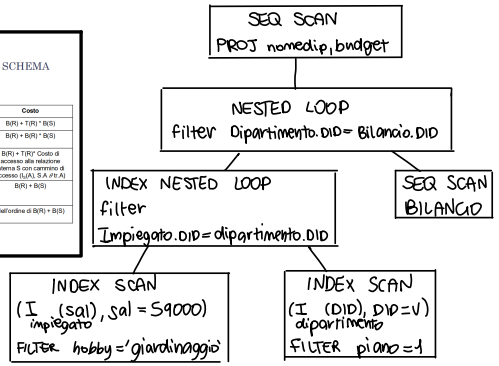
Stima della dimensione del risultato della selezione

- F(A=v) = 1/V(A,R)
- F(A IN (v1, v2,.....,vN))= N * F(A=v)
- F(A>v) = (Max(A,R) - v)/(Max(A,R)-Min(A,R))
- F(A<v) = (v-Min(A,R))/(Max(A,R)-Min(A,R))
- F(A1 = A2) = 1/ MAX (V(A1,R), V(A2,R))
- F(C1 AND C2) = F(C1) *F(C2)
- F(C1 OR C2) = F(C1) +F(C2) -F(C1) *F(C2)
- F(NOT C) = 1 - F(C)

5. Presentare un piano fisico di esecuzione che utilizzi un cammino di accesso indicizzato per almeno una relazione di base.

Operatori fisici per il join e schema fisico

Operatore fisico	Vincolo su schema fisico	Costo
Iterazione semplice	Nessuno	B(R) + T(R) * B(S)
Iterazione orientata ai blocchi	Nessuno	B(R) + B(R) * B(S)
Index nested loop	Indice su attributo di join della relazione inner	B(R) + T(R)* Costo di accesso alla relazione interna S con cammino di accesso (I$_A$), S A ∈tr A)
Merge join	Relazioni devono essere ordinate rispetto all'attributo di join	B(R) + B(S)
Hash join	Nessuno	Nell'ordine di B(R) + B(S)



Tipologia B

ESERCIZIO 1

Si considerino le relazioni con schema:

PIZZA (CodP, *Nome*, Costo)
CLIENTE(TelC, NomeC, Via, NCiv, NInt)
ORDINE(TelC^{CLIENTE}, Data, CodP^{PIZZA}, Qta)

si assuma inoltre che i cataloghi di sistema includano questo contenuto:

Relazioni

R	T(R)	B(R)
PIZZA	30	5
CLIENTE	500	100
ORDINE	10000	2500

Cluster

NomeC	Tipo	Su
C1	Index	PIZZA.Costo
C2	Hash	CLIENTE.TelC
C3	Hash	ORDINE.CodP

Indici

Chiave di ricerca	Tipo	H(I)	L(I)	CARD(I)
PIZZA.Costo	B+	2	5	20
PIZZA.Nome	B+	2	5	30
ORDINE.CodP	Hash			
ORDINE.Data	B+	2	30	300
ORDINE.TelC	B+	2	50	500
CLIENTE.TelC	Hash			

Attributi

A	R	V(A,R)	Min(A,R)	Max(A,R)
CodP	PIZZA	30		
Nome	PIZZA	30		
Costo	PIZZA	20	4	14
TelC	CLIENTE	500		
TelC	ORDINE	500		
Data	ORDINE	300	'01-Jan-2008'	'09-Jan-2009'
CodP	ORDINE	30		
Qta	ORDINE	15	1	20

Considerare l'interrogazione

SELECT NomeC, Via
FROM PIZZA NATURAL JOIN ORDINE NATURAL JOIN CLIENTE
WHERE Costo < 5 AND Data BETWEEN (DATE '4-Dec-2008' AND DATE '6-Dec-2008');

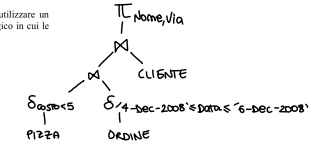
1. Determinare tutti i possibili cammini di accesso per le relazioni PIZZA, ORDINE e CLIENTE (considerare a questo proposito sia le condizioni di selezione sia le condizioni di join)

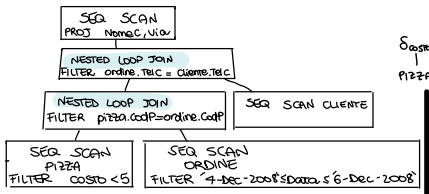
PIZZA:
- **Scansione sequenziale (vale sempre)**
- **Ipizza(COSTO), COSTO < 5**
- **L'attributo CodP è anche attributo di join con ORDINE, verifico se esiste indice su CodP. Non esiste, quindi non ci sono altri cammini di accesso**

ORDINE:
- **Scansione sequenziale**
- **Iordine(DATA), '4-Dec-2008' <= Data <= '6-Dec-2008'**
- **Iordine(CODP), codP=V con V attributo di join**
- **Iordine(TELC), telC=V con V attributo di join**

CLIENTE:
- **Scansione sequenziale**
- **Icliente(TELC), telC=V con V attributo di join**

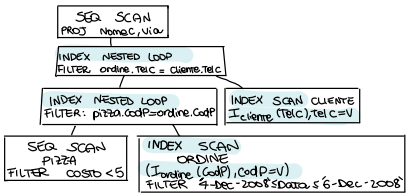
2. Presentare un esempio di piano di esecuzione fisico per l'interrogazione, supponendo di utilizzare un cammino di accesso sequenziale per tutte le relazioni, partendo dal piano di esecuzione logico in cui le selezioni vengono eseguite prima del join e la proiezione dopo.





Definizione di fattore booleano: il fattore booleano è una condizione atomica che, quando è falsa, rende falsa tutta la condizione di selezione.

3. Presentare, se esiste, un esempio di piano di esecuzione fisico per l'interrogazione, supponendo di utilizzare l'operatore di join index nested loop per entrambi i join.



Aggiornamento velocizzato: nel SET metto gli attributi senza indici, e nel WHERE quelli con indice non clusterizzato.

Aggiornamento rallentato: nel SET metto un attributo indicizzato, mentre il WHERE non contiene attributi indicizzati.

4. Presentare, se esiste, un esempio di piano di esecuzione fisico per l'interrogazione, supponendo di utilizzare l'operatore di merge join per entrambi i join senza aggiungere ulteriori nodi rispetto al piano logico.

Per utilizzare due merge join come operatori fisici per i due join, le tabelle in input devono essere ordinate rispetto all'attributo di join, quindi devono esistere i seguenti indici ordinati e clusterizzati: Ipizza(CodP), Iordine(CodP), Iordine(TelC), Iordine(TelC). Tuttavia l'indice Ipizza(CodP) non esiste e l'indice Iordine(TelC) non è clusterizzato. Quindi il piano non esiste.

5. Determinare se i piani presentati ai punti (3) e (4) ammettono elaborazione in pipelining.

Sintesi elaborazione degli operatori relazionali tramite operatori fisici

_____= ammette elaborazione in pipeline (nel join su relazione outer)

Operatore fisico			
Operatore logico	Iterazione	Uso di indici	Partizionamento
ordinamento		x	merge sort a due fasi
σ	x	x	
π^D	x		
π (con eliminazione duplicati)		x	Ordinamento (uso di funzione hash, non presentato)
⋈	nested loop semplice e a blocchi	index join	merge join hash join

Il piano al punto 3 ammette elaborazione in pipelining (la scansione sequenziale, il nested loop join e la proiezione senza eliminazione di duplicati ammettono elaborazione in pipelining).

Il piano al punto 4 ammette elaborazione in pipelining (la scansione sequenziale, l'index nested loop join e la proiezione senza eliminazione di duplicati ammettono elaborazione in pipelining).

6. I piani presentati ai punti (3) e (4) sono left deep?

Si, i piani sono left deep perché la relazione inner di ogni operatore di join (a destra per convenzione nel piano) è sempre una relazione di base.

ESERCIZIO 2

Si considerino le relazioni e lo schema fisico presentato nell'esercizio 1.
Considerare l'interrogazione

SELECT TelC
FROM ORDINE
WHERE Qta = 5 AND CodP = 2

1. Calcolare una stima del numero di tuple restituite dall'interrogazione.

Per prima cosa calcoliamo il fattore di selettività della condizione:
F(Qtà=5) = 1/V (Qta, ORDINE) = 1/15
F(CodP=2) = 1/V (CodP, ORDINE) = 1/30
F(Qtà=5 AND CodP=2) = F(Qtà=5)*F(Cod=2) = 1/15 * 1/30 = 1/450
Per ottenere il numero stimato di tuple restituite, moltiplichiamo il numero di tuple della relazione ORDINE per il valore di selettività.
T(ORDINE)*F(Qtà=5 AND CodP=2) = 10000/450 = 22,22 quindi circa 23 tuple

2. Determinare i fattori booleani. → **I fattori booleani sono Qta = 5 e CodP = 2**

3. Supponendo di non avere ancora creato alcun indice e di poterne creare uno solo, su quale attributo converrebbe crearlo? Discutere l'impatto della clusterizzazione su questo indice. [la parte in blu è un chiarimento rispetto al testo discusso a lezione]

Converrebbe crearlo sulla condizione più selettiva, quindi quella che restituisce meno tuple (cioè con il fattore di selettività più basso): CodP = 2.

Stima della dimensione del risultato della selezione

- F(A=v) = 1/V(A,R)
- F(A IN (v1, v2,.....,vN))= N * F(A=v)
- F(A>v) = (Max(A,R) - v)/(Max(A,R)-Min(A,R))
- F(A<v) = (v-Min(A,R))/(Max(A,R)-Min(A,R))
- F(A1 = A2) = 1/ MAX (V(A1,R), V(A2,R))
- F(C1 AND C2) = F(C1) *F(C2)
- F(C1 OR C2) = F(C1) +F(C2) -F(C1) *F(C2)
- F(NOT C) = 1 - F(C)

Tipologia C

ESERCIZIO 1

Data la seguente relazione:

MODELLO(codice, capacita', peso, anno_p, cod_produttore)

su cui sono definiti un indice clusterizzato su codice, un indice non clusterizzato su anno_p e un indice non clusterizzato su cod_produttore:

1. fornire un esempio di interrogazione che è velocizzata dall'avere l'indice su anno_p implementato come indice hash: uguaglianze

**SELECT *
FROM MODELLO
WHERE anno_p = 2004**

2. fornire un esempio di interrogazione che è velocizzata dall'avere l'indice su anno_p implementato come indice ordinato: uguaglianza o disuguaglianza

**SELECT *
FROM MODELLO
WHERE anno_p > 2004**

3. fornire un esempio di aggiornamento che è velocizzato dalla presenza di tali indici;

**UPDATE MODELLO
SET peso = 10
WHERE anno_p > 2007 AND codice = 111**
(WHERE)

Un aggiornamento viene velocizzato dalla presenza di indici se la clausola di qualificazione include almeno una condizione di selezione rispetto a un attributo indicizzato e la clausola SET non contiene attributi indicizzati.

4. fornire un esempio di aggiornamento che è rallentato dalla presenza di tali indici.

**UPDATE MODELLO
SET cod_produttore = 'ABC'
WHERE peso > 70**

Un aggiornamento viene rallentato dalla presenza di indici se la clausola SET riferisce almeno un attributo indicizzato e la clausola di qualificazione non contiene attributi indicizzati.

ESERCIZIO 2

Data la seguente relazione:

IMPIEGATO(imp#, nome, stip, età, dip#)

con un indice clusterizzato su imp# e un indice non clusterizzato su età:

1. fornire un esempio di interrogazione che è velocizzata dall'avere l'indice su imp# implementato come indice hash;
2. fornire un esempio di interrogazione che è velocizzata dall'avere l'indice su imp# implementato come indice ordinato;
3. fornire un esempio di interrogazione che è velocizzata dall'avere l'indice su età implementato come indice hash;
4. fornire un esempio di interrogazione che è velocizzata dall'avere l'indice su età implementato come indice ordinato;
5. la presenza degli indici velocizza il controllo del vincolo di chiave? in che casi?
6. fornire un esempio di operazione di aggiornamento rallentata dall'indice;
7. fornire un esempio di operazione di aggiornamento velocizzata dall'indice.

**1) SELECT *
FROM IMPIEGATO
WHERE IMP# = 12345**

**2) SELECT *
FROM IMPIEGATO
WHERE IMP# > 34500**

**3) SELECT *
FROM IMPIEGATO
WHERE età = 18**

**4) SELECT *
FROM IMPIEGATO
WHERE età BETWEEN 18 AND 30**

**5) La presenza di un indice su un attributo che è anche una chiave velocizza il controllo dell'unicità del valore. Nei casi in cui si inserisce un nuovo record o quando si aggiorna un record esistente, il database deve verificare che il valore per la chiave non sia già presente. Se esiste un indice sulla chiave, questa ricerca è molto più rapida.
Grazie all'indice, la ricerca del valore da inserire/modificare è molto più veloce rispetto ad una scansione completa di tutta la tabella.**

**6) UPDATE IMPIEGATO
SET età = età + 1
WHERE stip > 2000**

**7) UPDATE IMPIEGATO
SET stip = 3000
WHERE età > 30**

Normalizzazione

Esercizio I.B NORMALIZZAZIONE

Si consideri lo schema di relazione R(U,V,W,X,Y,Z) e il seguente insieme minimale di dipendenze funzionali:

- U→V
- VW→X
- X → YZ

1. Mostrare una relazione su tale schema che contenga almeno due tuple e verifichi le dipendenze, ed una relazione che contenga almeno due tuple e non le verifichi.

verifica

U	V	W	X	Y	Z
5	a	1	c	e	f
6	b	2	d	g	h

non verifica

U	V	W	X	Y	Z
5	a	1	c	e	f
5	b	2	d	g	h

la relazione non è verificata perchè 5→a nella seconda tupla non viene rispettata in quanto al posto di a abbiamo b

2. Determinare le chiavi della relazione.

$\{uw\}^+ = \{u,w,v,x,y,z\}$

Siccome la chiusura corrisponde allo schema di R,(U,W) è superchiave. È anche minimale e unica, pertanto l'unica chiave è (U,W)

3. Lo schema è in BCNF? È il 3NF? Giustificare le risposte.

Non è in 3NF perchè a sinistra della freccia, ad esempio su U→V, non abbiamo supechiave e a destra della freccia non abbiamo attributi primi.
Non è in BCNF perchè non abbiamo superchiavi a sinistra della freccia.

4. Mostrare una decomposizione non lossless join dello schema. Giustificare la risposta.

R1 (U,V,W)
R2 (X,Y,Z)

$R1 \cap R2 = \emptyset$

Poichè l'intersezione tra R1 e R2 non ritorna la chiave, la scomposizione non è lossless join

5. Mostrare una decomposizione dello schema che non preservi le dipendenze. Giustificare la risposta.

R1 (U,X)
R2 (V,W,X)
R3 (X,Y,Z)
non preserva le dipendenze perchè U→V viene persa

6. Mostrare una decomposizione dello schema usando l'algoritmo proposto. La decomposizione proposta è in BCNF? E' in 3NF? Preserva le dipendenze? È lossless join?

R1 (U,V) U→V
R2 (V,W,X) V,W→X
R3 (X,Y,Z) X→Y,Z
R4 (U,W)

* R1 (U,V)
R2 (V,W,X)
R3 (X,Y,Z)
R4 (U,W)

L'algoritmo garantisce che lo schema risultante sia in 3NF, lossless join e soddisfi le dipendenze
* Tutte le dipendenze hanno a sinistra una chiave, quindi lo schema è in BCNF

(ABC) chiave 3NF: $ABC \xrightarrow{o} X$, $X \rightarrow B$ BCNF: $ABC \rightarrow X$

È in 3NF se la dipendenza funzionale ha la chiave a sinistra della freccia o un attributo primo a destra della freccia. È BCNF se ha la chiave a sinistra della freccia.

Una decomposizione R è lossless join se l'intersezione tra R1 e R2 ritorna almeno una chiave. È quindi non lossless join se l'intersezione non ritorna una chiave.
Una decomposizione conserva tutte le dipendenze se, se si esegue la chiusura, ritorna tutti gli elementi della relazione originale R.

PRODOTTO (Prodotto, Componente, Tipo, Q, PC, Fornitore, PT)

Determinare, per ciascuna delle seguenti affermazioni, se rappresentano dipendenze funzionali per la relazione PRODOTTO e, in caso affermativo, presentare la dipendenza:

a) Ogni fornitore propone un unico prezzo unitario di vendita per ogni singola componente.

Fornitore, Componente → PT

b) Ci possono essere più componenti con lo stesso prezzo unitario.

non è dipendenza funzionale: più tuple con diversi componenti e stesso PT

c) Il tipo di ogni componente è unico.

Componente → Tipo

SENSORE (codS, modello, lunghezza, larghezza, altezza, posizione, parametro, valore, data, ora)

Determinare, per ciascuna delle seguenti affermazioni, se rappresentano dipendenze funzionali per la relazione SENSORE. In caso affermativo, presentare la dipendenza; in caso negativo, motivare opportunamente.

a) Lunghezza, larghezza e altezza sono unici per ciascun modello.

modello → lunghezza, larghezza, altezza

b) Ogni dispositivo, in una determinata data e una determinata ora, può rilevare un solo valore per un certo parametro.

CodS, data, ora, parametro → valore

c) Un dispositivo può essere spostato nel tempo.

uno a molti, più tuple con stesso dispositivo ma coordinate diverse

APPUNTAMENTO (IdPaziente, NomePaziente, IndirizzoPaziente, NomeMedico, Specialità, DataAppuntamento, OraAppuntamento)

Determinare, per ciascuna delle seguenti affermazioni, se rappresentano dipendenze funzionali per la relazione APPUNTAMENTO. In caso affermativo, presentare la dipendenza; in caso negativo, motivare opportunamente.

a) L'indirizzo di ogni paziente è unico.

IdPaziente → IndirizzoPaziente

b) Più di un medico può ricevere in uno stesso giorno alla stessa ora.

non è dipendenza funzionale: più tuple con stessa data e stessa ora, ora medico diversO

c) Un paziente può prenotare al più una visita al giorno per ogni specializzazione.

IdPaziente, DataAppuntamento, Specialità → OraAppuntamento

DRONE (codD, modello, raggio, autonomia, posizione, tipoMissione, area, data, ora)

Determinare, per ciascuna delle seguenti affermazioni, se rappresentano dipendenze funzionali per la relazione DRONE. In caso affermativo, presentare la dipendenza; in caso negativo, motivare opportunamente.

a) Raggio e autonomia sono fissi per ciascun modello.

modello → raggio, autonomia

b) Ogni drone, in una determinata data e ora, può svolgere una sola missione su una certa area e si può trovare in una sola posizione.

codD, data, ora → tipoMissione, area, posizione

c) Ogni missione può coprire più aree.

non si può fare (uno a molti)

Algebra Relazionale

Esercizio II.A – ALGEBRA RELAZIONALE

In riferimento al seguente schema relazionale

PERSONA(CF, Nome, Indirizzo, Città, Provincia, CAP, Telefono, DataNascita)
AVVOCATO (AvvocatoID, $CF^{PERSONA}$)
SPECIALIZZAZIONE(CodSpec, NomeSpec)
HA(AvvocatoID^{AVVOCATO}, CodSpec^{SPECIALIZZAZIONE})
ISCRIZIONE(AvvocatoID^{AVVOCATO}, NomeAlbo, Data)
TRIBUNALE(IdTrib, NomeTribunale, Città, Regione, CAP)
GIUDICE(GiudiceID, $CF^{PERSONA}$, AnniEsperienza, IdTrib^{TRIBUNALE})
CAUSA(IdTrib^{TRIBUNALE}, NumCausa, DescrizioneCausa, TipoCausa)
UDIENZA(IdTrib^{CAUSA}, NumCausa^{CAUSA}, NumUdienza, Data, Ora, Stanza, GiudiceID^{GIUDICE})
COINVOLGE(CF^{PERSONA}, NumCausa^{CAUSA}, IdTrib^{CAUSA}, RuoloP)
RAPPRESENTA(AvvocatoID^{AVVOCATO}, CF^{PERSONA}, NumCausa^{CAUSA}, IdTrib^{CAUSA}, RuoloA)

Formulare le seguenti interrogazioni in **algebra relazionale**. *Si suggerisce di verificare che i vincoli di schema siano rispettati e che lo schema del risultato corrisponda a quanto richiesto dall'interrogazione.*

1. Determinare codice fiscale, nome, indirizzo e data di nascita delle persone che sono coinvolte in cause (diverse) sia con ruolo "parte civile" che con ruolo "imputato"

$\pi_{CF,nome,indirizzo,DataNascita}(PERSONA \bowtie \pi_{CF}(\delta_{RuoloP='parte\ civile'}(COINVOLGE)) \cap \pi_{CF}(\delta_{RuoloP='Imputato'}(COINVOLGE)))$

2. Determinare gli avvocati genovesi in possesso di **tutte** le specializzazioni.

$\pi_{AvvocatoId,CodSpec}(\delta_{città='Genova'}(PERSONA) \bowtie AVVOCATO \bowtie HA) \div \pi_{CodSpec}(SPECIALIZZAZIONE)$

Esercizio II.A – ALGEBRA RELAZIONALE

In riferimento al seguente schema relazionale

PERSONA(CF, Nome, Indirizzo, Città, Provincia, CAP, Telefono, DataNascita)
AVVOCATO (AvvocatoID, $CF^{PERSONA}$)
SPECIALIZZAZIONE(CodSpec, NomeSpec)
HA(AvvocatoID^{AVVOCATO}, CodSpec^{SPECIALIZZAZIONE})
ISCRIZIONE(AvvocatoID^{AVVOCATO}, NomeAlbo, Data)
TRIBUNALE(IdTrib, NomeTribunale, Città, Regione, CAP)
GIUDICE(GiudiceID, $CF^{PERSONA}$, AnniEsperienza, IdTrib^{TRIBUNALE})
CAUSA(IdTrib^{TRIBUNALE}, NumCausa, DescrizioneCausa, TipoCausa)
UDIENZA(IdTrib^{CAUSA}, NumCausa^{CAUSA}, NumUdienza, Data, Ora, Stanza, GiudiceID^{GIUDICE})
COINVOLGE(CF^{PERSONA}, NumCausa^{CAUSA}, IdTrib^{CAUSA}, RuoloP)
RAPPRESENTA(AvvocatoID^{AVVOCATO}, CF^{PERSONA}, NumCausa^{CAUSA}, IdTrib^{CAUSA}, RuoloA)

Formulare le seguenti interrogazioni in **algebra relazionale**. *Si suggerisce di verificare che i vincoli di schema siano rispettati e che lo schema del risultato corrisponda a quanto richiesto dall'interrogazione.*

1. Determinare le persone rappresentate in una causa da avvocati specializzati sia in diritto penale che in diritto di famiglia.

$\pi_{CF}(RAPPRESENTA \bowtie \pi_{AvvocatoID}(\delta_{NomeSpec='diritto\ penale'}(SPECIALIZZAZIONE) \bowtie HA) \cap \pi_{AvvocatoID}(\delta_{NomeSpec='diritto\ di\ famiglia'}(SPECIAL) \bowtie HA))$

2. Determinare i giudici con meno di 20 anni di esperienza che hanno presieduto udienze in tutti i tribunali della Lombardia.

$\pi_{GiudiceID,IdTrib}(\pi_{GiudiceID}(\delta_{AnniEsperienza<=20}(GIUDICE) \bowtie UDIENZA)) \div \pi_{IDtrib}(\delta_{regione=Lombardia}(TRIBUNALE))$

Esercizio II.A – ALGEBRA RELAZIONALE

In riferimento al seguente schema relazionale, relativo alla selezione e alla gestione dei tutor in un'università:

STUDENTE(Matricola, Nome, Cognome, Genere, DataN, Nazionalità, CdS, Anno, CFU, Media)
POSIZIONE (IdP, *Tipo, Scuola, Progetto*, Descr, NumOre, NumPosti)
CORSO(IdC, Titolo, Ore, Modalità)
RICHIEDE(IdP^{POSIZIONE}, IdC^{CORSO})
DOMANDA(Matricola^{STUDENTE}, IdP^{POSIZIONE}, DataOra)
COLLOQUIO(Matricola^{STUDENTE}, Punteggio)
ASSEGNAZIONE(Matricola^{STUDENTE}, IdP^{POSIZIONE}, Ore, DataFirma$_O$)
COMPLETA(Matricola^{STUDENTE}, IdC^{CORSO}, DataOra)

Formulare le seguenti interrogazioni in **algebra relazionale**. *Si suggerisce di verificare che i vincoli di schema siano rispettati e che lo schema del risultato corrisponda a quanto richiesto dall'interrogazione.*

1. Determinare nome, cognome e matricola degli studenti che non hanno fatto domanda per alcuna posizione

$\pi_{name,cognome,matricola}(STUDENTE \bowtie (\pi_{matricola}(STUDENTE) - \pi_{matricola}(DOMANDA)))$

2. *(*)* Determinare le matricole degli studenti assegnati a qualche posizione che hanno completato un solo corso

$\pi_{matricola}(COMPLETA \bowtie ASSEGNAZIONE) - \pi_{matricola}(\rho_{idc \leftarrow idc2}(COMPLETA) \bowtie \delta_{idc \neq idc2}(COMPLETA))$

studenti che hanno completato un qualsiasi corso \ studenti che hanno completato più di un corso

Esercizio II.A – ALGEBRA RELAZIONALE

In riferimento al seguente schema relazionale, relativo a tornei di tennis:

GIOCATORE(IdG, Nome, Cognome, Genere, DataN, Nazione)
TORNEO(IdT, NomeT, Luogo, Datal, DataF, NumTurni, Tipo, Terreno)
CATEGORIA(IdCat, NomeCategoria, GenereCategoria)
REGISTRAZIONE(IdT^{TORNEO}, IdCat^{CATEGORIA}, NumRegistrazione, DataRegistrazione, TestaDiSerie)
GIOCAIN(IdT^{TREGISTRAZIONE}, IdCat^{REGISTRAZIONE} NumRegistrazione^{REGISTRAZIONE}, IdG^{GIOCATORE})
NOTA: un giocatore nel caso di singoli, due nel caso di doppi, più di due nel caso di squadre (categorie)

Formulare le seguenti interrogazioni in **algebra relazionale**.

1. I tornei a cui hanno partecipato sia giocatori italiani che francesi nella stessa categoria.

$\pi_{IdT}(\pi_{IdT,IdC}(\delta_{nazione='Italia' \wedge Genere='M'}(GIOCATORE) \bowtie GIOCAIN) \cap \pi_{IdT,IdC}(\delta_{nazione='Francia' \wedge Genere='M'}(GIOCATORE) \bowtie GIOCAIN))$

2. I giocatori che hanno partecipato a tornei su tutti i tipi di terreno

$\pi_{IdG,terreno}(\delta_{Genere='M'}(GIOCATORE) \bowtie GIOCAIN \bowtie TORNEO) \div \pi_{terreno}(TORNEO)$

Esercizio II.A – ALGEBRA RELAZIONALE

In riferimento al seguente schema relazionale relativo a tornei di tennis:

GIOCATORE(IdG, Nome, Cognome, Genere, DataN, Nazione)
TORNEO(IdT, NomeT, Luogo, Datal, DataF, NumTurni, Tipo, Terreno)
CATEGORIA(IdCat, NomeCategoria, GenereCategoria)
REGISTRAZIONE(IdT^{TORNEO}, IdCat^{CATEGORIA}, NumRegistrazione, DataRegistrazione, TestaDiSerie)
GIOCAIN(IdT^{TREGISTRAZIONE}, IdCat^{REGISTRAZIONE} NumRegistrazione^{REGISTRAZIONE}, IdG^{GIOCATORE})
NOTA: un giocatore nel caso di singoli, due nel caso di doppi, più di due nel caso di squadre (categorie)

1. I/Le giocatori/rici francesi che hanno partecipato sia a tornei su erba che su terra battuta (attributo Terreno) nello stesso anno

$\pi_{IdG}(\delta_{nazione='Francia'}(GIOCATORE)) \cap \pi_{IdG}(\pi_{IdG,YEAR(DataI)}(GIOCAIN \bowtie \delta_{Terreno='erba'}(TORNEO)) \cap (\pi_{IdG,YEAR(DataI)}(GIOCAIN \bowtie \delta_{Terreno='terra\ battuta'}(TORNEO)))$

2. Determinare i nomi dei soggetti che hanno utilizzato per le loro spedizioni tutte le categorie disponibili (cioè sono stati mittenti di almeno una spedizione con quella categoria, per ogni categoria).

$\pi_{nome}(\pi_{name,CodiceSp,CodiceCat}(SPEDIZIONE \bowtie_{mittente=CodiceSogg} SOGGETTO) \div \pi_{CodiceCat}(CATEGORIA))$

3. Determinare i/werdi dei clienti che hanno effettuato due prenotazioni diverse con la stessa data ritiro e agenzie di ritiro in città diverse.

$\pi_{email}(CLIENTE \bowtie (\delta_{città1 \neq città2}(\pi_{IdCli,DataRitiro,Città1}(PRENOTAZIONE \bowtie_{AgRitiro=NomeAg} AGENZIA) \bowtie \delta_{Città1=città2}(\pi_{IdCli,DataRitiro,Città2}(PRENOTAZIONE \bowtie_{AgRitiro=NomeAg} AGENZIA)))))$

a) Determinare i nomi dei creatori dei sondaggi che hanno sia opzioni (=quando) in orari pomeridiani (cioè con OraI > 13) che mattutini (cioè OraF < 13)

$\pi_{Nome}(PARTECIPANTE \bowtie_{IdP=IdCreatore} (SONDAGGIO) \bowtie (\pi_{IdD}(\delta_{OraF>13}(QUANDO)) \cap \pi_{IdD}(\delta_{OraF<13}(QUANDO))))$

SQL

Esercizio II.B - SQL

In riferimento al seguente schema relazionale, relativo alla selezione e alla gestione dei tutor in un'università:

STUDENTE(Matricola, Nome, Cognome, Genere, DataN, Nazionalità, CdS, Anno, CFU, Media)
POSIZIONE (IdP, *Tipo, Scuola, Progetto*, Descr, NumOre, NumPosti)
CORSO(IdC, Titolo, Ore, Modalità)
RICHIEDE(IdP^{POSIZIONE}, IdC^{CORSO})
DOMANDA(Matricola^{STUDENTE}, IdP^{POSIZIONE}, DataOra)
COLLOQUIO(Matricola^{STUDENTE}, Punteggio)
ASSEGNAZIONE(Matricola^{STUDENTE}, IdP^{POSIZIONE}, Ore, DataFirma$_O$)
COMPLETA(Matricola^{STUDENTE}, IdC^{CORSO}, DataOra)

Formulare le seguenti interrogazioni in **SQL**.

1. *(*)* Determinare matricola, nome e cognome di tutti gli studenti, con il punteggio del colloquio se disponibile, altrimenti NULL.

```
SELECT S.matricola, S.Nome, S.cognome C.Punteggio
FROM STUDENTE S LEFT OUTER JOIN COLLOQUIO C ON S.Matricola = C.Matricola
```

2. Determinare l'identificatore della posizione per cui sono richiesti più corsi

```
SELECT IdP
FROM RICHIEDE
GROUP BY IdP
HAVING COUNT (*) >= ALL (SELECT COUNT(*)
                         FROM RICHIEDE
                         GROUP BY IdP)
```

Esercizio II.B - SQL

Si consideri il seguente schema relazionale relativo alla selezione e alla gestione dei tutor in un'università

STUDENTE(Matricola, Nome, Cognome, Genere, DataN, Nazionalità, CdS, Anno, CFU, Media)
POSIZIONE (IdP, *Tipo, Scuola, Progetto*, Descr, NumOre, NumPosti)
CORSO(IdC, Titolo, Ore, Modalità)
RICHIEDE(IdP^{POSIZIONE}, IdC^{CORSO})
DOMANDA(Matricola^{STUDENTE}, IdP^{POSIZIONE}, DataOra)
ASSEGNAZIONE(Matricola^{STUDENTE}, IdP^{POSIZIONE}, Ore, DataFirma$_O$)
COMPLETA(Matricola^{STUDENTE}, IdC^{CORSO}, DataOra)
REGISTRO(Matricola^{STUDENTE}, Data, NumOre, Descrizione, NumPartecipanti$_O$)

Formulare le seguenti interrogazioni in **SQL**.

1. *(*)* Per tutte le posizioni mostrare l'id e i corsi richiesti, anche se non richiede alcun corso (NULL se nessuno)

```
SELECT IdP, IdC
FROM POSIZIONE LEFT OUTER NATURAL JOIN RICHIEDE
```

2. Determinare la matricola dello studente che ha registrato il maggior numero totale di ore nel registro.

```
SELECT Matricola
FROM REGISTRO
GROUP BY Matricola
HAVING SUM(NumOre) >= ALL (SELECT SUM(NumOre)
                           FROM REGISTRO
                           GROUP BY Matricola)
```

Esercizio II.B - SQL

In riferimento al seguente schema relazionale:

PERSONA(CF, Nome, Indirizzo, Città, Provincia, CAP, Telefono, DataNascita)
AVVOCATO (AvvocatoID, $CF^{PERSONA}$)
SPECIALIZZAZIONE(CodSpec, NomeSpec)
HA(AvvocatoID^{AVVOCATO}, CodSpec^{SPECIALIZZAZIONE})
ISCRIZIONE(AvvocatoID^{AVVOCATO}, NomeAlbo, Data)
TRIBUNALE(IdTrib, NomeTribunale, Città, Regione, CAP)
GIUDICE(GiudiceID, $CF^{PERSONA}$, AnniEsperienza, IdTrib^{TRIBUNALE})
CAUSA(IdTrib^{TRIBUNALE}, NumCausa, DescrizioneCausa, TipoCausa)
UDIENZA(IdTrib^{CAUSA}, NumCausa^{CAUSA}, NumUdienza, Data, Ora, Stanza, GiudiceID^{GIUDICE})
COINVOLGE(CF^{PERSONA}, NumCausa^{CAUSA}, IdTrib^{CAUSA}, RuoloP)
RAPPRESENTA(AvvocatoID^{AVVOCATO}, CF^{PERSONA}, NumCausa^{CAUSA}, IdTrib^{CAUSA}, RuoloA)

Formulare le seguenti interrogazioni in **SQL**.

1. Determinare le persone coinvolte in una causa per cui non è presente l'avvocato che le rappresenta in quella causa

```
SELECT DISTINCT CF
FROM COINVOLGE
WHERE (CF, IdTrib, NumCausa) NOT IN (SELECT (CF, IdTrib, NumCausa)
                                     FROM RAPPRESENTA)
```

oppure

```
SELECT DISTINCT CF
FROM COINVOLGE LEFT OUTER NATURAL JOIN RAPPRESENTA
WHERE AvvocatoID IS NULL
```

2. Determinare i giudici il cui numero di anni di esperienza è superiore alla media degli anni di esperienza dei giudici appartenenti allo stesso tribunale

```
SELECT G.GiudiceID
FROM GIUDICE G
WHERE G.AnniEsperienza > (SELECT AVG(AnniEsperienza)
                          FROM GIUDICE
                          WHERE IdTrib = G.AnniEsperienza)
```

Esercizio II.B - SQL

In riferimento al seguente schema relazionale:

PERSONA(CF, Nome, Indirizzo, Città, Provincia, CAP, Telefono, DataNascita)
AVVOCATO (AvvocatoID, $CF^{PERSONA}$)
SPECIALIZZAZIONE(CodSpec, NomeSpec)
HA(AvvocatoID^{AVVOCATO}, CodSpec^{SPECIALIZZAZIONE})
ISCRIZIONE(AvvocatoID^{AVVOCATO}, NomeAlbo, Data)
TRIBUNALE(IdTrib, NomeTribunale, Città, Regione, CAP)
GIUDICE(GiudiceID, $CF^{PERSONA}$, AnniEsperienza, IdTrib^{TRIBUNALE})
CAUSA(IdTrib^{TRIBUNALE}, NumCausa, DescrizioneCausa, TipoCausa)
UDIENZA(IdTrib^{CAUSA}, NumCausa^{CAUSA}, NumUdienza, Data, Ora, Stanza, GiudiceID^{GIUDICE})
COINVOLGE(CF^{PERSONA}, NumCausa^{CAUSA}, IdTrib^{CAUSA}, RuoloP)
RAPPRESENTA(AvvocatoID^{AVVOCATO}, CF^{PERSONA}, NumCausa^{CAUSA}, IdTrib^{CAUSA}, RuoloA)

Formulare le seguenti interrogazioni in **SQL**.

1. Determinare i giudici assegnati a un tribunale che non hanno ancora presieduto udienze in quel tribunale

```
SELECT DISTINCT GiudiceID
FROM GIUDICE
WHERE (GiudiceID, IdTrib) NOT IN (SELECT (GiudiceID, IdTrib)
                                  FROM UDIENZA)
```

oppure

```
SELECT DISTINCT GiudiceID
FROM GIUDICE LEFT OUTER NATURAL JOIN UDIENZA
WHERE NumCausa IS NULL
```

2. Determinare gli avvocati la cui anzianità di iscrizione[1] a un albo è superiore alla media di anzianità di iscrizione degli avvocati iscritti allo stesso albo.

```
SELECT I.AvvocatoID
FROM ISCRIZIONE I
WHERE CURRENT_DATE - I.Data > (SELECT AVG(CURRENT_DATE - Data)
                               FROM ISCRIZIONE
                               WHERE NomeAlbo = I.NomeAlbo)
```

Esercizio II.B - SQL

In riferimento al seguente schema relazionale relativo a tornei di tennis:

GIOCATORE(IdG, Nome, Cognome, Genere, DataN, Nazione)
TORNEO(IdT, NomeT, Luogo, Datal, DataF, NumTurni, Tipo, Terreno)
CATEGORIA(IdCat, NomeCategoria, GenereCategoria)
REGISTRAZIONE(IdT^{TORNEO}, IdCat^{CATEGORIA}, NumRegistrazione, DataRegistrazione, TestaDiSerie)
GIOCAIN(IdT^{TREGISTRAZIONE}, IdCat^{REGISTRAZIONE} NumRegistrazione^{REGISTRAZIONE}, IdG^{GIOCATORE})
NOTA: un giocatore nel caso di singoli, due nel caso di doppi, più di due nel caso di squadre (categorie)

Formulare le seguenti interrogazioni in **SQL**.

1. I/Le giocatori/rici tedeschi/e che non hanno mai partecipato a Wimbledon

```
SELECT DISTINCT IdG
FROM GIOCATORE
WHERE Nazione = 'Germania'
AND IdG NOT IN (SELECT DISCTINCT IdG
                FROM GIOCAIN NATURAL JOIN TORNEO
                WHERE NomeT = 'Wimbledon')
```

2. Il torneo con l'età media dei giocatori più alta

```
SELECT DISTINCT IdT
FROM TORNEO NATURAL JOIN GIOCAIN NATURAL JOIN GIOCATORE
GROUP BY IdT
HAVING AVG(DataN - Datal) >= ALL (SELECT AVG(DataN - Datal)
                                  FROM TORNEO NATURAL JOIN
                                       GIOCAIN NATURAL JOIN GIOCATORE
                                  GROUP BY IdT)
```

Esercizio II.B - SQL

In riferimento al seguente schema relazionale, relativo ad assicurazioni sanitarie e alle relative pratiche per il rimborso delle prestazioni coperte dall'assicurazione

POLIZZA(NumP, Titolare^{TITOLARE}, DataIn, DataScad, Tipo, ImportoP)
PRATICA(CodicePr, NumP^{POLIZZA}, DataPratica)
PRESTAZIONE(CodicePr^{PRATICA}, NumRiga, Erogante^{SOGGETTO}, Beneficiario^{SOGGETTO}, Descrizione, DataPrest, ImportoPrest)
ASSISTITO(CodiceFisc, Nome, Cognome, Genere, DataN)
SOGGETTO(CodiceSogg, Denominazione, Indirizzo)
TITOLARE(Titolare^{ASSISTITO}, *Email*, Telefono, Indirizzo, Convenzionato)
FAMILIARE(Familiare^{ASSISTITO}, Titolare^{TITOLARE})
COPRE_FAM(NumP^{POLIZZA}, Familiare^{FAMILIARE})

Formulare le seguenti interrogazioni in **SQL**.

1. Determinare la data della pratica, il numero di polizza e nome e cognome del titolare della polizza per le pratiche presentate nell'ultimo mese che non contengono prestazioni di tipo visita (cioè tali che Descrizione include la stringa 'visita').

```
SELECT DataPratica, NumP, Nome, Cognome
FROM PRATICA NATURAL JOIN POLIZZA JOIN ASSISTITO ON Titolare=CodiceFisc
WHERE DataPratica >= CURRENT_DATE - INTERVAL 1 MONTH
AND DataPratica <= CURRENT_DATE AND CodicePr NOT IN (SELECT CodicePr
                                                     FROM PRESTAZIONE
                                                     WHERE DESCRIZIONE
                                                     LIKE '%visita%'
```

Esercizio II.B - SQL

In riferimento al seguente schema relazionale:

DIPENDENTE(ID, Congnome, Nome, Dipartimento^{DIPARTIMENTO})
PROFESSORE(ID^{DIPENDENTE}, Qualifica, AnnoNascita)
DIPARTIMENTO(Codice, *Nome*, Indirizzo, Direttore^{PROFESSORE})
CORSODISTUDIO(Codice, *Nome*, Dipartimento^{DIPARTIMENTO})
COLLABORAZIONE(CorsoDiStudio^{CORSODISTUDIO}, Professore^{PROFESSORE}, Tipo)
CORSO(Codice, Materia^{MATERIA}, Docente^{PROFESSORE}, CorsoDiStudio^{CORSODISTUDIO}, Semestre)
MATERIA(Sigla, *Nome*)

Formulare le seguenti interrogazioni in **SQL**.

1. Determinare per ogni dipartimento di direttore, il numero di dipendenti e il numero di corsi di studio gestiti dal dipartimento.

```
SELECT Dipartimento, Direttore, COUNT(DISTINCT ID), COUNT(DISTINCT CORSODISTUDIO.Codice)
FROM DIPENDENTE JOIN CORSODISTUDIO USING (Dipartimento) JOIN DIPARTIMENTO ON
                Dipartimento=DIPARTIMENTO.Codice
GROUP BY Dipartimento, Direttore
```