# Esame di SQL

*Punteggi massimi:*

- *Domande 1 e 2 svolte perfettamente: 23;*
- *Domande 1 e 3 svolte perfettamente: 25;*
- *Domande 2 e 3 svolte perfettamente: 28;*
- *Domande 1, 2 e 3 svolte perfettamente: 33.*

*Lo svolgimento corretto di una sola domanda non permette il raggiungimento della sufficienza.*

Le seguenti relazioni definiscono una base di dati "**Prenotazioni**" per gestire le prenotazioni di camere in un hotel di Torino. Gli attributi sottolineati sono le chiavi primarie delle relazioni.

CLIENTE(NumTessera, Nome, DataNascita, CittàResidenza)
CAMERA(Num, Piano, NumPostiLetto, Tipo, Prezzo)
PRENOTA(Cliente, Num, Piano, DataArrivo, DataPartenza)

Vincoli di integrità referenziali: PRENOTA(Cliente) referenzia CLIENTE(NumTessera) e PRENOTA(Num,Piano) referenzia CAMERA(Num,Piano).
Significato degli attributi: "NumTessera" è il numero tessera del cliente accreditato; "Tipo" può assumere i valori "economica", "normale" e "suite"; "Prezzo" è il prezzo di una notte in euro. I rimanenti attributi sono autoesplicativi. Gli attributi sono tutti NOT NULL.
**Commentare ogni soluzione proposta spiegando le varie parti della query.**

**Domanda 1 (bassa complessità).**
Con riferimento alla base di dati "Prenotazioni" esprimere in SQL la seguente interrogazione:
**Trovare i clienti che hanno almeno due prenotazioni di camere con un numero diverso di posti letto.**

**Soluzione 1.**

```
SELECT DISTINCT p1.Cliente
FROM prenota p1 JOIN prenota p2 ON (p1.Cliente=p2.Cliente)
     JOIN camera c1 ON (p1.Num=c1.Num AND p1.Piano=c1.Piano)
     JOIN camera c2 ON (p2.Num=c2.Num AND p2.Piano=c2.Piano)
WHERE c1.NumPostiLetto<>c2.NumPostiLetto;
```

La query effettua un self join tra coppie di prenotazioni con i relativi dati delle camere per ricavare i clienti che hanno effettuato almeno due prenotazioni. La condizione di join tra p1 e p2 assicura che ogni coppia di prenotazioni appartenga allo stesso cliente. La condizione nella where garantisce che la coppia di prenotazioni abbia un diverso numero di posti letto escludendo quindi le coppie composte da una stessa prenotazione. La distinct evita di dare in output più volte lo stesso cliente sia nel caso abbia effettuato più prenotazioni di questo tipo sia per il fatto che, a causa della simmetria della condizione nella clausola where.

**Domanda 2 (media complessità).**
Con riferimento alla base di dati "Prenotazioni" esprimere in SQL la seguente interrogazione:
**Mostrare i clienti che pagano, a notte, un prezzo maggiore del prezzo medio pagato, a notte, dai clienti provenienti dalla loro stessa città. Non usare la clausola WITH.**

**Soluzione 2.**

```
SELECT DISTINCT cl.NumTessera, cl.Nome, cl.CittàResidenza
FROM cliente cl JOIN prenota p ON (cl.Cliente=p.NumTessera)
     JOIN camera ca ON (p.Num=ca.Num AND p.Piano=ca.Piano)
WHERE ca.Prezzo > (SELECT AVG(Prezzo)
                   FROM cliente cl1 JOIN prenota p1 ON (cl1.Cliente=p1.NumTessera)
                        JOIN camera ca1 ON (ca1.Num=p1.Num AND ca1.Piano=p1.Piano)
                    WHERE cl1.CittàResidenza=cl.CittàResidenza);
```

Con la superquery consideriamo i clienti con le relative prenotazioni e i dati delle camere e con la sottoquery

correlata ricaviamo il prezzo medio delle camere prenotate dai clienti della stessa città del cliente considerato nella superquery. La condizione where della superquery assicura che nel risultato abbiamo solo i clienti che rispettano la condizione della consegna. Come richiesto, non abbiamo usato la clausola with.

**Domanda 3 (alta complessità).**
Con riferimento alla base di dati "Prenotazioni" esprimere in SQL la seguente interrogazione:
**Trovare, SENZA utilizzare gli operatori insiemistici, quanti clienti hanno sempre e soltanto prenotato camere di tipo economico.**

**Soluzione 3.**

```
SELECT COUNT(DISTINCT Cliente)
FROM prenota p
WHERE NOT EXISTS
          (SELECT *
           FROM prenota p1 JOIN camera c1 ON (p1.Num=c1.Num AND p1.Piano=c1.Piano)
           WHERE p1.Cliente=p.Cliente AND c1.Tipo<>'economica');
```

L'uso degli avverbi "sempre" e "soltanto" suggerisce il fatto che la query deve usare una negazione.
Con la superquery consideriamo tutte le prenotazioni e con la sottoquery ricaviamo le prenotazioni di un certo cliente di camere non economiche, cioè rispondiamo alla domanda negata.
La condizione NOT EXISTS nella superquery permette di escludere tali clienti e cioè di ricavare i clienti che hanno prenotato solo camere economiche.
Grazie all'operatore aggregato COUNT(DISTINCT ) contiamo quanti sono questi clienti evitando di considerare i duplicati dati dai clienti che hanno effettuato più prenotazioni.
Si noti che, se nella superquery avessimo ricavato i clienti partendo dalla relazione CLIENTE invece che dalla relazione PRENOTA, avremmo potuto ottenere nel risultato clienti che non hanno mai effettuato nessuna prenotazione.
Invece una query come

```
SELECT COUNT(DISTINCT Cliente)
FROM prenota p JOIN camera c ON (p.Num=c.Num AND p.Piano=c.Piano)
WHERE c.Tipo='economica';
```

NON risponde alla consegna perché conta anche i clienti che hanno effettuato più prenotazioni sia di camere economiche che non economiche.

# Esame di Teoria

**Domanda 1 (9 punti).**
Con riferimento alla base di dati "**Prenotazioni**":

A. Esprimere in Algebra Relazionale l'interrogazione
**Elencare i clienti che hanno prenotato in ogni piano in cui è presente una suite.**

B. Esprimere, nel calcolo relazionale su tuple con dichiarazione di range, la seguente domanda:
**Elencare le camere con un solo posto letto che sono in overbooking. Si mostri Num e Piano.**
**(Versione più semplice: Una camera in overbooking è una camera assegnata a clienti diversi con stessa data di arrivo).**
**(Versione più difficile: Una camera in overbooking è una camera assegnata a clienti diversi per periodi che coincidono anche solo in parte).**

**Soluzione 1.**
**A.** Una possibile soluzione in algebra relazionale è la seguente:

$$R(A,B) \div S(B)$$

dove

$$R(A,B) = \pi_{Cliente,\ Piano}(PRENOTA)$$
$$S(B) = \pi_{Piano}(\sigma_{Tipo='suite'}(CAMERA))$$

**B.** Una possibile soluzione è la seguente:

Versione più semplice:

$$\{p.Num, p.Piano \mid p(PRENOTA) \mid \exists c(CAMERA)(c.NumPostiLetto=1 \wedge p.Num=c.Num \wedge p.Piano=c.Piano \wedge \exists p'(PRENOTA)\ (p.Num=p'.Num \wedge p.Piano=p'.Piano \wedge p.Cliente\neq p'.Cliente \wedge p.DataArrivo=p'.DataArrivo))\}$$

Versione più difficile:

$$\{p.Num, p.Piano \mid p(PRENOTA) \mid \exists c(CAMERA)(c.NumPostiLetto=1 \wedge p.Num=c.Num \wedge p.Piano=c.Piano \wedge \exists p'(PRENOTA)\ (p.Num=p'.Num \wedge p.Piano=p'.Piano \wedge p.Cliente\neq p'.Cliente \wedge (p.DataArrivo \leq p'.DataArrivo < p.DataPartenza \vee p'.DataArrivo \leq p.DataArrivo < p'.DataPartenza))\}$$

**Domanda 2 (8 punti).**
A. Dare la definizione di insieme di copertura minimale.
B. Dati: R( L, M, N, O, P, Q, R) e F = { O → M R, N → L M Q, M N → O P, R → N O }
dire se R è in 3FN e se non lo è decomporla in relazioni in 3FN esplicitando tutti i passaggi. Il risultato è BCNF?

**Soluzione 2.**
A. Si vedano gli appunti/testo/slide.

B. Per prima cosa è necessario identificare le chiavi candidate. In questo caso abbiamo tre chiavi K1={O}, K2={N}, K3={R} (il calcolo della chiusura lo dimostra).

La relazione R è già in 3FN e in BCNF, in quanto tutte le dipendenze funzionali sono del tipo "superchiave". Non è necessario decomporre.

**Domanda 3 (8 punti).**
Si consideri la base di dati col seguente schema:

AUTORE(Nome, Nazionalità, Qualifica)
ARTICOLO(Titolo, Anno, Conferenza)
PUBBLICAZIONE(TitoloArticolo, NomeAutore)
dove TitoloArticolo e NomeAutore in PUBBLICAZIONE sono in vincolo di chiave esterna, rispettivamente, con ARTICOLO e AUTORE.

e i seguenti dati quantitativi:
CARD(pubblicazione) = 100 000
VAL(NomeAutore,pubblicazione) = 2 000
CARD(articolo) = 6 000
VAL(Nazionalità,autore) = 150
MIN(Anno, articolo) = 1972
MAX(Anno, articolo) = 2022

disegnare gli alberi sintattici prima e dopo l'ottimizzazione logica e calcolare il numero di tuple "mosse" prima e dopo l'ottimizzazione logica della seguente query:

$$\sigma_{Anno\geq 2002 \wedge Anno\leq 2022 \wedge NomeAutore='Alan\ Turing'}(articolo \bowtie_{Titolo=TitoloArticolo} pubblicazione)$$

**Soluzione 3.**
La query ottimizzata dividendo la selezione e portandola verso le foglie è:

$$\sigma_{Anno\geq 2002 \wedge Anno\leq 2022}(articolo) \bowtie_{Titolo=TitoloArticolo} \sigma_{NomeAutore='Alan\ Turing'}(pubblicazione)$$

Prima dell'ottimizzazione:
• Costo r1 = $articolo \bowtie_{Titolo=TitoloArticolo} pubblicazione$: $6 \cdot 10^3 \cdot 10^5 = 6 \cdot 10^8$.
• Cardinalità |r1| = CARD(pubblicazione) = $10^5$ (equijoin attraverso la chiave esterna).
• Costo della selezione = |r1| = $10^5$.
• Costo totale = $6 \cdot 10^8 + 10^5 \cong 6 \cdot 10^8$.

Dopo l'ottimizzazione:
• Costo $\sigma_1 = \sigma_{Anno\geq 2002 \wedge Anno\leq 2022}(articolo)$= CARD(articolo) = $6 \cdot 10^3$.
• Tuple prodotte dalla selezione $|\sigma_1|$ = (2022 – 2002)/(MAX(Anno, articolo) – MIN(Anno, articolo)) · CARD(articolo) = (2022 – 2002)/(2022 – 1972) · $6 \cdot 10^3$= 20/50 x $6 \cdot 10^3$= $2{,}4 \cdot 10^3$
• Costo $\sigma_2 = \sigma_{NomeAutore='Alan\ Turing'}(pubblicazione)$ = CARD(pubblicazione) = $10^5$.
• Tuple prodotte dalla selezione $|\sigma_2|$ = 1/VAL(NomeAutore,pubblicazione) · CARD(pubblicazione) = $\frac{1}{2000} \cdot 10^5 = 0{,}5 \cdot 10^2$.
• Costo join r = $\sigma_1 \bowtie_{Titolo=TitoloArticolo} \sigma_2 = 2{,}4 \cdot 10^3 \cdot 0{,}5 \cdot 10^2 = 1{,}2 \cdot 10^5$.
• Costo totale = $6 \cdot 10^3 + 10^5 + 1{,}2 \cdot 10^5 \cong 2 \cdot 10^5$.

**Domanda 4 (8 punti).**
Considerare la seguente storia interfogliata S = r1(x), r1(y), r2(x), r3(y), w1(y), r2(y), w2(x)
S è compatibile con il protocollo 2PL? Giustificare la risposta.

**Soluzione 4.**
È compatibile con 2PL perché, per esempio, è possibile aggiungere i lock ottenendo la storia seguente, compatibile con 2PL:

S = LS1(x), r1(x), LS1(y), r1(y), LS2(x), r2(x), LS3(y), r3(y), UN3(y), LX1(y), w1(y), UN1(x), UN1(y), LS2(y), r2(y), LX2(x), w2(x), UN2(y), UN2(x)