CATEGORIA 1: QUERY "ALMENO N" - Pattern Fondamentale

Algebra Relazionale - Template De Leoni

Pattern base per "almeno 2":

```
πA(R ⋈A=A1∧B≠B1 ρA1,B1←A,B(R))
```

Pattern per "almeno 3":

```
S1 := ρA1,B1←A,B(R)
S2 := ρA2,B2←A,B(R)
S3 := R × S1 × S2
πA(σA=A1∧A=A2∧B≠B1∧B≠B2∧B1≠B2(S3))
```

Esempi Tipici degli Esami De Leoni

1. "Clienti che hanno acquistato almeno due prodotti diversi"

```
ACQUISTO(CF_Cliente, Prodotto, Data)

ALGEBRA:
C1 := ACQUISTO
C2 := ACQUISTO
πCF_Cliente(C1 ⋈C1.CF_Cliente=C2.CF_Cliente ∧ C1.Prodotto≠C2.Prodotto C2)

SQL:
SELECT DISTINCT a1.CF_Cliente
FROM ACQUISTO a1, ACQUISTO a2
WHERE a1.CF_Cliente = a2.CF_Cliente
  AND a1.Prodotto != a2.Prodotto;
```

2. "Persone che frequentano almeno due biblioteche diverse"

```
FREQUENTAZIONE(CFPersona, CodiceBiblio, Data)

ALGEBRA:
```

```
F1 := FREQUENTAZIONE
F2 := FREQUENTAZIONE
πF1.CFPersona(F1 ⋈F1.CFPersona=F2.CFPersona ∧
F1.CodiceBiblio≠F2.CodiceBiblio F2)

SQL:
SELECT DISTINCT f1.CFPersona
FROM FREQUENTAZIONE f1
JOIN FREQUENTAZIONE f2 ON f1.CFPersona = f2.CFPersona
WHERE f1.CodiceBiblio != f2.CodiceBiblio;
```

CATEGORIA 2: QUERY "ESATTAMENTE N" - Pattern Differenza

Template De Leoni

```
ALMENO_N − ALMENO_N+1
```

Esempio Tipico: "Esattamente Un Unico"

"Codici fiscali dei clienti che nel 2023 hanno prenotato in esattamente un unico stabilimento"

```
PRENOTAZIONE(CF_Cliente, IdStabilimento, Data)

ALGEBRA:
P1 := σData≥'1/1/2023' ∧ Data≤'31/12/2023'(PRENOTAZIONE)
P2 := P1
P3CF := πP1.CF_Cliente(P1 ⋈P1.CF_Cliente=P2.CF_Cliente ∧
P1.IdStabilimento≠P2.IdStabilimento P2)
RISULTATO := πCF_Cliente(P1) − P3CF

SQL:
SELECT p1.CF_Cliente
FROM PRENOTAZIONE p1
WHERE p1.Data BETWEEN '2023-01-01' AND '2023-12-31'
  AND NOT EXISTS (
    SELECT * FROM PRENOTAZIONE p2
    WHERE p2.CF_Cliente = p1.CF_Cliente
      AND p2.IdStabilimento != p1.IdStabilimento
```

```
      AND p2.Data BETWEEN '2023-01-01' AND '2023-12-31'
  );
```

CATEGORIA 3: QUERY "PER TUTTI" / DIVISIONE

Template Algebra - Due Varianti De Leoni

Variante 1: Divisione Classica

```
πA,B(R) ÷ πB(S)
```

Variante 2: Doppia Negazione (Preferita da De Leoni)

```
πA(R) − πA((πA(R) × πB(S)) − πA,B(R))
```

Esempi Tipici

1. "Studenti che hanno passato tutti gli esami"

```
ESAME(Studente, Corso, Voto)
CORSO(Corso, Docente)

ALGEBRA (Doppia Negazione):
S1 := πStudente,Corso(ESAME)
S2 := πCorso(CORSO)
πStudente(S1) − πStudente((πStudente(S1) × S2) − S1)

SQL con NOT EXISTS:
SELECT DISTINCT e1.Studente
FROM ESAME e1
WHERE NOT EXISTS (
  SELECT * FROM CORSO c
  WHERE NOT EXISTS (
    SELECT * FROM ESAME e2
    WHERE e2.Studente = e1.Studente
      AND e2.Corso = c.Corso
  )
);
```

2. "Clienti che hanno acquistato tutti i prodotti"

```
ACQUISTO(CF_Cliente, Prodotto)
PRODOTTO(Prodotto, Nome)

SQL con COUNT:
SELECT a.CF_Cliente
FROM ACQUISTO a
GROUP BY a.CF_Cliente
HAVING COUNT(DISTINCT a.Prodotto) = (SELECT COUNT(*) FROM PRODOTTO);
```

CATEGORIA 4: QUERY "SOLO" / NEGAZIONE

Pattern De Leoni per "Solo X"

```
TUTTI_A − A_CHE_HANNO_ANCHE_NON_X
```

Esempio Tipico

"Delegati che hanno partecipato solo a meeting non italiani"

```
DELEGATO(IdDelegato, Nome, Cognome)
PARTECIPA(IdDelegato, IdMeeting)
MEETING(IdMeeting, Nazione)

ALGEBRA:
DELEGATI_DA_TENERE := πIdDelegato(DELEGATO) −
πIdDelegato(σNazione='Italia'(PARTECIPA ⋈ MEETING))
πNome,Cognome(DELEGATI_DA_TENERE ⋈ DELEGATO)

SQL:
SELECT d.Nome, d.Cognome
FROM DELEGATO d
WHERE d.IdDelegato NOT IN (
  SELECT p.IdDelegato
  FROM PARTECIPA p
  JOIN MEETING m ON p.IdMeeting = m.IdMeeting
  WHERE m.Nazione = 'Italia'
);
```

CATEGORIA 5: QUERY MINIMO/MASSIMO

Template De Leoni

Massimo Assoluto:

```
πB(R) − πB(R ⋈B<B1 ρA1,B1←A,B(R))
```

Massimo Relativo (per ogni A):

```
πA,B(R) − πA,B(R ⋈A=A1∧B<B1 ρA1,B1←A,B(R))
```

Esempio Tipico

"Tipo di aereo con il massimo numero di passeggeri tra quelli in uso per voli che partono da Roma"

```
VOLO(Id, CittàPart, TipoAereo)
AEREO(TipoAereo, NumPasseggeri)

ALGEBRA:
C1 := σCittàPart='Roma'(VOLO ⋈ AEREO)
C2 := C1
TIPO_AEREO_NON_MASSIMO := πC1.TipoAereo(C1
⋈C1.NumPasseggeri<C2.NumPasseggeri C2)
πTipoAereo(C1) − TIPO_AEREO_NON_MASSIMO

SQL:
SELECT DISTINCT v.TipoAereo
FROM VOLO v
JOIN AEREO a ON v.TipoAereo = a.TipoAereo
WHERE v.CittàPart = 'Roma'
  AND a.NumPasseggeri = (
    SELECT MAX(a2.NumPasseggeri)
    FROM VOLO v2
    JOIN AEREO a2 ON v2.TipoAereo = a2.TipoAereo
    WHERE v2.CittàPart = 'Roma'
  );
```

CATEGORIA 6: QUERY CON AGGREGAZIONI - SQL SPECIFICO

Pattern COUNT con HAVING

Template per "più di N":

```
SELECT attributo, COUNT(*)
FROM tabella
GROUP BY attributo
HAVING COUNT(*) > N;
```

Pattern COUNT DISTINCT

Per contare elementi diversi:

```
SELECT cliente
FROM ordini
GROUP BY cliente
HAVING COUNT(DISTINCT prodotto) = (SELECT COUNT(*) FROM prodotti);
```

Esempi Specifici De Leoni

1. "Dipartimenti con più di 10 impiegati"

```
SELECT dipartimento, COUNT(*) AS num_impiegati
FROM IMPIEGATO
GROUP BY dipartimento
HAVING COUNT(*) > 10;
```

2. "Clienti che hanno ordinato tutti i prodotti della categoria 'Elettronica'"

```
SELECT c.cliente_id
FROM ORDINE o
JOIN PRODOTTO p ON o.prodotto_id = p.id
WHERE p.categoria = 'Elettronica'
GROUP BY c.cliente_id
HAVING COUNT(DISTINCT p.id) = (
  SELECT COUNT(*)
  FROM PRODOTTO
  WHERE categoria = 'Elettronica'
);
```

CATEGORIA 7: NOT IN vs EXISTS vs NOT EXISTS

Quando Usare Cosa negli Esami De Leoni

NOT IN: Quando cerchi elementi NON presenti in un insieme

```
SELECT nome
FROM STUDENTE
WHERE id NOT IN (SELECT studente_id FROM ESAME WHERE voto < 18);
```

NOT EXISTS: Per negazioni complesse e divisioni

```
SELECT s.nome
FROM STUDENTE s
WHERE NOT EXISTS (
  SELECT * FROM CORSO c
  WHERE NOT EXISTS (
    SELECT * FROM ESAME e
    WHERE e.studente_id = s.id AND e.corso_id = c.id
  )
);
```

EXISTS: Per verificare esistenza con condizioni

```
SELECT s.nome
FROM STUDENTE s
WHERE EXISTS (
  SELECT * FROM ESAME e
  WHERE e.studente_id = s.id AND e.voto >= 27
);
```

METODOLOGIA D'ESAME PER L'ESERCIZIO 3

Step 1: Analisi della Richiesta (2 minuti)

1. Identifica il tipo di query:
   - "Almeno N" → Autojoin
   - "Esattamente N" → Differenza
   - "Tutti" → Divisione
   - "Solo" → Negazione
   - "Massimo/Minimo" → Confronto

Step 2: Algebra Relazionale (8 minuti)

1. Usa variabili temporanee (S1, S2, ecc.)
2. Applica il template appropriato
3. Verifica condizioni di join
4. Controlla proiezioni finali

Step 3: SQL (10 minuti)

1. Traduci l'algebra in SQL
2. Usa alias chiari
3. Testa mentalmente su casi limite
4. Scegli tra EXISTS/NOT IN in base al caso

Step 4: Verifica (2 minuti)

1. Rileggi la richiesta
2. Controlla cardinalità attesa
3. Verifica sintassi SQL

ERRORI FATALI DA EVITARE

In Algebra Relazionale:

❌ **Dimenticare ridenominazioni** nei theta-join ❌ **Proiezioni sbagliate** (proiettare attributi non necessari) ❌ **Condizioni di join incomplete** negli autojoin

In SQL:

❌ **GROUP BY senza aggregazioni** necessarie ❌ **HAVING invece di WHERE** per filtri sui dati ❌ **NOT IN con valori NULL** (usa NOT EXISTS) ❌ **COUNT(*) invece di COUNT(DISTINCT)** quando servono valori unici

Ricorda:

- **L'esercizio 3 è SBARRAMENTO** - devi prenderlo bene
- **Mostra sempre il ragionamento**, anche se sbagli la sintassi
- **Usa variabili temporanee** per chiarezza
- **Traduci sempre "tutti" in doppia negazione** negli esami di De Leoni

Esempi pratici

Si consideri la seguente base di dati per la registrazione dei concorsi, i candidati e gli esiti:

- CANDIDATO(CF, Nome, Cognome)
- PARTECIPA(CF, CodConcorso, Esito)
- CONCORSO(CodConcorso, Descrizione, Anno)

dove Esito può essere 'positivo' o 'negativo' (usare queste due costanti).

A. Nel riquadro, scrivere una Query in Algebra Relazione che restituisce i nomi e cognomi di tutti i candidati con esito negativo per almeno un concorso del 2019 (2 punti).[2]

```
P1 = σ(Anno = 2019 AND Esito = "Negativo") [ PARTECIPA ⋈ CONCORSO ]
P2 = P1

π(P1.Nome, Cognome) ⋈ (P1.CodConcorso = P2.CodConcorso AND P1.CF = P2.CF)
```

$$\pi_{Nome,Cognome}(CANDIDATO \bowtie (\sigma_{Esito='negativo' \text{ AND } Anno=2019}(CONCORSO \bowtie PARTECIPA)))$$

Si consideri la seguente base di dati per la registrazione dei concorsi, i candidati e gli esiti:

- CANDIDATO(CF, Nome, Cognome)
- PARTECIPA(CF, CodConcorso, Esito)
- CONCORSO(CodConcorso, Descrizione, Anno)

B. Nel riquadro, scrivere un'interrogazione SQL che restituisca tutti i codici dei concorsi che hanno avuto solo esiti negativi, senza duplicati (2.5 punti).

```sql
SELECT DISTINCT CodConcorso
FROM Partecipa P
WHERE CodConcorso NOT IN (SELECT CodConcorso
                          FROM Partecipa
                          WHERE Esito = "Positivo");
```

```
SELECT CodConcorso FROM CONCORSO
EXCEPT
SELECT CodConcorso FROM PARTECIPA WHERE
Esito='Positivo'

    oppure

SELECT CodConcorso FROM CONCORSO
WHERE CodConcorso NOT IN

    (SELECT CodConcorso FROM PARTECIPA
    WHERE Esito='Positivo')
```

Si consideri la seguente base di dati per la registrazione dei concorsi, i candidati e gli esiti:

- CANDIDATO(CF, Nome, Cognome)
- PARTECIPA(CF, CodConcorso, Esito)
- CONCORSO(CodConcorso, Descrizione, Anno)

C. Nel Riquadro, scrivere un'interrogazione SQL che restituisca il codice fiscale dei candidati che hanno partecipato a TUTTI i concorsi con descrizione "Banca di Italia" a partire dal 2000 (2.5 punti)

```sql
SELECT CF
FROM Partecipa
WHERE CodConcorso IN (SELECT CodConcorso
            FROM Partecipa P, Concorso C
            WHERE P.CodConcorso = C.CodConcorso
            AND Descrizione = "Banca d'Italia"
            AND Anno > 2000);

OPPURE

CREATE VIEW Tutti_concorsi (Numero, CF)
SELECT COUNT(*), CF
FROM Partecipa P, Concorso C
WHERE P.CodConcorso = C.CodConcorso
AND Descrizione = "Banca d'Italia"
AND Anno > 2000
GROUP BY CF;

SELECT T1.CF
FROM Tutti_concorsi T1, Tutti_concorsi T2
WHERE T1.CF = T2.CF
```

```
AND T1.Numero = T2.Numero;
```

```
SELECT CF
FROM PARTECIPA P, CONCORSO C
WHERE P.CodConcorso = C CodConcorso AND
Descrizione='Banca di Italia' AND Anno>2000
GROUP BY CF
HAVING COUNT(*) =
    (SELECT COUNT(*) FROM CONCORSO
    WHERE Descrizione='Banca di Italia'
    AND Anno>2000)
```