Guida Completa Basi di Dati - Prof. De Leoni UniPD

1. MODELLAZIONE ER - CASI CRITICI E TRANELLI

1.1 Entità vs Attributi - Decisioni Cruciali

Regola di De Leoni: Se qualcosa ha proprietà proprie che interessano, è un'entità.

Esempi critici:

- **Indirizzo**: Entità se interessa via, CAP, città separatamente; Attributo se è solo stringa
- **Telefono**: Entità se distinguo tipo (casa/ufficio/mobile); Attributo se è solo numero
- **Data**: Sempre attributo, mai entità

1.2 Cardinalità - I Pattern Tipici degli Esami

Pattern 1: Impresa e Sedi

- Impresa (1,1) ---- (1,N) Sede
- "Ogni sede appartiene a una sola impresa, ogni impresa ha almeno una sede"

Pattern 2: Procedimento e Soggetti

- Procedimento (1,1) ---- (0,1) Persona (relativa a "referente")
- Procedimento (1,N) ---- (0,N) Soggetto (relazione generica "relativo a")

Pattern 3: Generalizzazioni con Vincoli

- Persona → {Dipendente, Cliente} (Totale, Esclusiva)
- Documento → {Lettera, Fax} (Parziale, Esclusiva)

1.3 Identificatori Esterni - Casi Difficili

Regola: Entità debole quando l'identificazione DEVE passare attraverso un'altra entità.

Esempio tipo esame:

```
Studente ----- Iscrizione ----- Corso
    (1,N)         (1,1)         (1,N)
```

- Iscrizione identificata da: Studente + Corso + AnnoAccademico

2. ALGEBRA RELAZIONALE - PATTERN RICORRENTI NEGLI ESAMI

2.1 Divisione (Query "Per Tutti")

Template base:

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

Esempio del corso: "Studenti che hanno passato TUTTI gli esami"

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

πStudente(ESAME) - πStudente((πStudente(ESAME) × πCorso(CORSO)) - πStudente,Corso(ESAME))
```

2.2 Query "Almeno N" - Autojoin con Conteggio

Pattern: Trovare entità associate ad almeno N altre entità

Esempio: "Clienti con almeno 3 ordini"

```
πClienti(ORDINE) ./Cliente1=Cliente2 ∧ Ordine1≠Ordine2
(ρClienti1,Ordine1←Cliente,Ordine(ORDINE)) ./Cliente2=Cliente3 ∧ Ordine2≠Ordine3
(ρClienti2,Ordine2←Cliente,Ordine(ORDINE)) ./...
```

2.3 Confronti Temporali - Pattern de Leoni

"Ultimi/Primi per categoria":

```
πA,B,data(R) - πA,B,data1(R ./A1=A ∧ B1=B ∧ data1<data ρA1,B1,data1←A,B,data(R))
```

2.4 Negazioni Complesse

Pattern: "A che non hanno B con proprietà P"

```
πA(R) - πA(σP(R ./condizione S))
```

3. NORMALIZZAZIONE - METODOLOGIA SISTEMATICA

3.1 Algoritmo Completo per Determinare Chiavi

```
1. Calcolare X+ per ogni sottoinsieme X di attributi
2. Identificare superchiavi: X tale che X+ = tutti gli attributi
3. Eliminare superchiavi non minimali
4. Le rimanenti sono chiavi candidate
```

3.2 Verifiche Forme Normali - Checklist De Leoni

1NF: ✓ Domini atomici **2NF**: ✓ 1NF + ✓ Nessuna dipendenza parziale da chiave **3NF**: ✓ 2NF + ✓ Nessuna dipendenza transitiva **BCNF**: ✓ 3NF + ✓ Ogni determinante è superchiave

3.3 Decomposizione BCNF - Algoritmo degli Esami

```
Dato schema R(A1,...,An) con dipendenze F:
1. Se R è in BCNF, STOP
2. Trova dipendenza X→Y che viola BCNF
3. Decompongi in R1(X,Y) e R2(R-Y)
4. Ripeti ricorsivamente su R1 e R2
```

Verifica senza perdita: R1 ∩ R2 deve essere chiave in R1 o R2

3.4 Casi Tranello degli Esami

Tranello 1: Dipendenze implicite

- A→B, B→C implica A→C (transitività)
- A→BC è equivalente a A→B e A→C (decomposizione)

Tranello 2: Attributi che non compaiono in dipendenze

- Sempre fanno parte di ogni chiave candidata

4. INDICI - METODOLOGIA DI SCELTA

4.1 Regole Ferree per Query con WHERE

```sql
SELECT * FROM R WHERE X=val AND Y>val2 ORDER BY Z
```

Ordine indice composto:

1. Attributi di uguaglianza (X)
2. Attributi di range (Y)
3. Attributi di ordinamento (Z)

Indice ottimale: B+Tree su (X,Y,Z)

4.2 Decisioni Hash vs B+Tree

- **Hash**: SOLO uguaglianza (=)
- **B+Tree**: Uguaglianza + Range + ORDER BY

4.3 Esempi Tipici degli Esami

Query 1: `SELECT * FROM R WHERE A=4 AND B>8` **Risposta**: B+Tree su (A,B) - A primo (uguaglianza), B secondo (range)

Query 2: `SELECT * FROM R WHERE C='valore'` **Risposta**: Hash su C (solo uguaglianza)

5. SQL - COSTRUTTI AVANZATI E TRANELLI

5.1 Query Nidificate - Pattern De Leoni

EXISTS per divisione:

```sql
SELECT s.nome FROM Studenti s
WHERE NOT EXISTS (
    SELECT * FROM Corsi c
    WHERE NOT EXISTS (
        SELECT * FROM Esami e
        WHERE e.studente = s.id AND e.corso = c.id
    )
);
```

5.2 Funzioni Aggregate con GROUP BY

Regola: Con GROUP BY, SELECT può contenere SOLO:

- Attributi del GROUP BY
- Funzioni aggregate
- Costanti

Errore comune:

```sql
-- SBAGLIATO
SELECT nome, COUNT(*) FROM Studenti GROUP BY corso;

-- CORRETTO
SELECT corso, COUNT(*) FROM Studenti GROUP BY corso;
```

5.3 HAVING vs WHERE

- **WHERE**: Filtra righe PRIMA del raggruppamento
- **HAVING**: Filtra gruppi DOPO l'aggregazione

```sql
SELECT corso, AVG(voto)
FROM Esami
WHERE voto >= 18          -- Considera solo voti sufficienti
GROUP BY corso
HAVING COUNT(*) > 10;     -- Solo corsi con >10 esami
```

6. TRANSAZIONI - TEORIA E PRACTICE

6.1 Proprietà ACID - Definizioni Precise

- **Atomicity**: Tutto o niente
- **Consistency**: Da stato consistente a stato consistente
- **Isolation**: Come se fossero seriali
- **Durability**: Effetti permanenti dopo commit

6.2 Serializzabilità - Test del Grafo

Algorithm:

1. Costruisci grafo delle precedenze
2. Arco Ti → Tj se Ti precede Tj in conflitto
3. Schedule serializzabile ⟺ Grafo aciclico

6.3 Deadlock - Prevenzione e Rilevamento

Wait-Die: Ti aspetta Tj solo se timestamp(Ti) < timestamp(Tj) **Wound-Wait**: Ti uccide Tj solo se timestamp(Ti) < timestamp(Tj)

7. RIPRISTINO - LOG E PROTOCOLLI

7.1 Write-Ahead Logging (WAL)

Regole:

1. Log record deve essere scritto prima della modifica DB
2. Tutti i log della transazione prima del COMMIT

7.2 REDO vs UNDO - Decisioni

- **REDO**: Transazioni COMMITTED dopo ultimo checkpoint
- **UNDO**: Transazioni ACTIVE al momento del crash

8. PATTERN DI PROGETTO - SCHEMI RICORRENTI

8.1 Gestione Storico

```
CONTRATTO(id, cliente, data_inizio, data_fine, ...)
```

- data_fine = NULL per contratti attivi

8.2 Gerarchie Organizzative

```
DIPENDENTE(id, nome, manager_id REFERENCES DIPENDENTE(id))
```

8.3 Cataloghi con Varianti

```
PRODOTTO(id, nome, categoria)
VARIANTE(prodotto_id, colore, taglia, prezzo)
```

9. ERRORI COMUNI E COME EVITARLI

9.1 Modellazione ER

❌ **Sbagliato**: Mettere identificatori in relazioni molti-a-molti ✅ **Corretto**: Identificatori solo su entità

❌ **Sbagliato**: Attributi derivabili (età da data_nascita)
✅ **Corretto**: Solo attributi necessari e non derivabili

9.2 Algebra Relazionale

❌ **Sbagliato**: Proiezione prima di selezione quando necessario attributo per selezione ✅ **Corretto**: Selezione prima, poi proiezione

9.3 SQL

❌ **Sbagliato**: GROUP BY senza aggregazione quando non necessario ✅ **Corretto**: GROUP BY solo se serve aggregazione

10. STRATEGIE D'ESAME

10.1 Gestione del Tempo

- **ER (30 min)**: Identifica entità, poi relazioni, poi cardinalità
- **Algebra (20 min)**: Traduci richiesta in italiano, poi formalizza
- **Normalizzazione (25 min)**: Chiavi prima, poi verifiche sistematiche
- **SQL/Indici (15 min)**: Pattern recognition

10.2 Controlli di Consistenza

- **ER**: Ogni entità ha almeno un attributo identificativo
- **Normalizzazione**: Verifica conservazione dipendenze
- **SQL**: Test su casi limite (insiemi vuoti, valori NULL)

10.3 Trucchi per Recuperare Punti

- Anche se non sai la risposta completa, mostra il ragionamento
- Disegna schemi parziali se bloccato
- Scrivi SQL anche se non perfetto, meglio di niente
- In normalizzazione, elenca sempre le dipendenze trovate

Questa guida copre tutto il programma di De Leoni con focus sui pattern specifici che compaiono nei suoi esami. La chiave è riconoscere i pattern e applicare metodologie sistematiche.