Indice

1. Modellazione ER
2. Progettazione Logica
3. Algebra Relazionale
4. SQL
5. Normalizzazione
6. Transazioni e Ripristino
7. Indici e Ottimizzazione
8. Quiz Tipici

---

1. Modellazione ER

1.1 Elementi principali del modello ER

Entità

- Rappresentano classi di oggetti con proprietà comuni
- Ogni istanza è univocamente identificabile

Attributi

- Proprietà che descrivono un'entità o relazione
- Tipi: semplici, compositi, derivati, multivalore
- Identificatori: sottolineati nel diagramma

Relazioni

- Collegamenti logici tra due o più entità
- Caratterizzate da cardinalità
- Possono avere attributi propri

Cardinalità

- `(0,1)`: partecipazione opzionale, al massimo un'occorrenza
- `(1,1)`: partecipazione obbligatoria, esattamente un'occorrenza
- `(0,N)`: partecipazione opzionale, più occorrenze possibili
- `(1,N)`: partecipazione obbligatoria, almeno un'occorrenza

Generalizzazione

- Totale (freccia piena): ogni istanza dell'entità padre deve essere un'istanza di almeno una delle entità figlie
- Parziale (freccia vuota): un'istanza dell'entità padre può non appartenere a nessuna entità figlia
- Esclusiva (default): un'istanza dell'entità padre appartiene a una sola entità figlia
- Sovrapposta: un'istanza dell'entità padre può appartenere a più entità figlie

1.2 Casi particolari e vincoli

Vincoli di identificazione esterna

- Un'entità viene identificata attraverso la sua relazione con un'altra entità
- Rappresentata con una freccia dall'entità alla relazione

Attributi multivalore

- Rappresentati con ellisse con bordo doppio
- Possono assumere più valori per una singola entità

Entità deboli

- Non hanno identificatori propri
- Dipendono da un'entità forte per l'identificazione

Auto-relazioni

- Relazioni di un'entità con se stessa
- Ruoli: specificano la funzione dell'entità nella relazione

1.3 Metodologia di progettazione

1. **Identificare le entità principali**
2. **Definire gli attributi** per ciascuna entità
3. **Stabilire le relazioni** tra le entità
4. **Definire le cardinalità** delle relazioni
5. **Identificare generalizzazioni/specializzazioni**
6. **Verificare l'assenza di ridondanze**
7. **Verificare la presenza degli identificatori** per ogni entità

1.4 Errori comuni da evitare

- Utilizzare relazioni quando servono entità (es. per eventi, transazioni)

- Dimenticare gli identificatori
- Definire cardinalità errate
- Confondere generalizzazioni esclusive e sovrapposte
- Rappresentare come attributo ciò che dovrebbe essere un'entità

1.5 Esempi di modellazione ER

Esempio 1: Sistema bibliotecario

```
LIBRO(Codice, Titolo, Anno, Editore)
AUTORE(CF, Nome, Cognome)
PRESTITO(Data inizio, Data fine, Data restituzione effettiva)
```

Con relazioni:

- SCRITTO_DA tra LIBRO e AUTORE (N:N)
- PRENDE_IN_PRESTITO tra UTENTE e LIBRO (1:N) che diventa l'entità PRESTITO

Esempio 2: Sistema bancario

```
CLIENTE(CF, Nome, Cognome, Indirizzo)
CONTO(Numero, Saldo, DataApertura)
FILIALE(Codice, Indirizzo, Città)
```

Con relazioni:

- INTESTATO tra CLIENTE e CONTO (N:N)
- GESTITO_DA tra CONTO e FILIALE (N:1)

---

2. Progettazione Logica

2.1 Ristrutturazione del modello ER

Eliminazione delle generalizzazioni

- **Accorpamento verso il padre**: eliminare entità figlie e aggiungere i loro attributi all'entità padre con un attributo discriminante
- **Accorpamento verso le figlie**: eliminare entità padre e copiare i suoi attributi nelle entità figlie
- **Sostituzione con relazioni**: trasformare la generalizzazione in relazioni tra entità padre e figlie

Eliminazione attributi multivalore

- Creazione di una nuova entità per l'attributo multivalore
- Relazione 1:N con l'entità originaria

Eliminazione relazioni N:N

- Trasformare in entità con relazioni 1:N con le entità originarie

Eliminazione relazioni complesse (con più di 2 entità)

- Trasformare in entità con relazioni binarie

2.2 Conversione in modello relazionale

Regole di traduzione

1. **Entità forti**:

```
ENTITÀ(attributi, identificatore)
```

2. **Entità deboli**:

```
ENTITÀ_DEBOLE(attributi, chiave_esterna → ENTITÀ_FORTE)
```

3. **Relazioni 1:1**:

```
Inserire chiave esterna in una delle due tabelle:
ENTITÀ1(attributi, chiave_esterna → ENTITÀ2)
```

4. **Relazioni 1:N**:

```
Inserire chiave esterna nella tabella dal lato N:
ENTITÀ_N(attributi, chiave_esterna → ENTITÀ_1)
```

5. **Relazioni N:N**:

```
Creare nuova tabella per la relazione:
RELAZIONE(chiave_esterna1 → ENTITÀ1, chiave_esterna2 → ENTITÀ2,
attributi_relazione)
```

6. **Generalizzazioni**:

- **Verso il padre**:

```
ENTITÀ_PADRE(attributi_padre, attributi_figlie, tipo)
```

- **Verso le figlie**:

```
ENTITÀ_FIGLIA1(attributi_padre,
attributi_figlia1)ENTITÀ_FIGLIA2(attributi_padre, attributi_figlia2)
```

- **Con relazioni**:

```
ENTITÀ_PADRE(attributi_padre)ENTITÀ_FIGLIA1(attributi_figlia1,
chiave_esterna → ENTITÀ_PADRE)ENTITÀ_FIGLIA2(attributi_figlia2,
chiave_esterna → ENTITÀ_PADRE)
```

2.3 Vincoli di integrità

1. **Vincoli di dominio**: valori ammissibili per un attributo
2. **Vincoli di chiave**: unicità di una combinazione di attributi
3. **Vincoli di integrità referenziale**: coerenza tra chiavi esterne e chiavi primarie
4. **Vincoli generali**: condizioni arbitrarie sui dati

2.4 Esempio di ristrutturazione e traduzione

Modello ER originale:

- PERSONA(CF, Nome, Cognome, DataNascita)
- CLIENTE e DIPENDENTE come specializzazioni di PERSONA
- CLIENTE(NumeroTelefono)
- DIPENDENTE(Stipendio, DataAssunzione)

Ristrutturazione (accorpamento verso il padre):

```
PERSONA(CF, Nome, Cognome, DataNascita, Tipo, NumeroTelefono, Stipendio,
DataAssunzione)
```

Con vincoli:

- Se Tipo = 'Cliente', allora NumeroTelefono NOT NULL, Stipendio NULL, DataAssunzione NULL

- Se Tipo = 'Dipendente', allora NumeroTelefono NULL, Stipendio NOT NULL, DataAssunzione NOT NULL

Schema relazionale finale:

```
PERSONA(CF, Nome, Cognome, DataNascita, Tipo, NumeroTelefono, Stipendio,
DataAssunzione)
```

Dove:

- CF è chiave primaria
- NumeroTelefono, Stipendio, DataAssunzione possono essere NULL

---

3. Algebra Relazionale

3.1 Operatori fondamentali

Selezione (σ)

- **Sintassi**: $\sigma_{condizione}(R)$
- **Funzione**: Filtra tuple secondo una condizione
- **Esempio**: $\sigma_{Età>25}(PERSONA)$

Proiezione (π)

- **Sintassi**: $\pi_{attributi}(R)$
- **Funzione**: Seleziona colonne e rimuove duplicati
- **Esempio**: $\pi_{Nome,Cognome}(PERSONA)$

Join naturale (⋈)

- **Sintassi**: R ⋈ S
- **Funzione**: Combina tuple con valori uguali su attributi comuni
- **Esempio**: STUDENTE ⋈ CORSO

Theta-join ($\bowtie_\theta$)

- **Sintassi**: $R \bowtie_{condizione} S$
- **Funzione**: Combina tuple che soddisfano una condizione
- **Esempio**: $IMPIEGATO \bowtie_{Impiegato.Dipartimento=Dipartimento.ID} DIPARTIMENTO$

Prodotto cartesiano (×)

- **Sintassi**: R × S
- **Funzione**: Combina ogni tupla di R con ogni tupla di S
- **Esempio**: PERSONA × CITTÀ

Unione (∪)

- **Sintassi**: R ∪ S
- **Funzione**: Restituisce tuple in R o S o entrambe
- **Condizione**: R e S devono avere lo stesso schema

Differenza (-)

- **Sintassi**: R - S
- **Funzione**: Restituisce tuple in R ma non in S
- **Esempio**: IMPIEGATO - MANAGER

Intersezione (∩)

- **Sintassi**: R ∩ S
- **Funzione**: Restituisce tuple sia in R che in S

Divisione (÷)

- **Sintassi**: R ÷ S
- **Funzione**: Trova tuple di R che si combinano con tutte le tuple di S

3.2 Pattern comuni negli esami

1. Trovare elementi che soddisfano una condizione "almeno n volte"

```
// Clienti che hanno acquistato almeno due prodotti diversi
A1 = ACQUISTO
A2 = ACQUISTO
π_CF_Cliente(A1 ⋈_(A1.CF_Cliente=A2.CF_Cliente AND A1.Prodotto≠A2.Prodotto)
A2)
```

2. Trovare elementi che soddisfano una condizione "esattamente n volte"

```
// Clienti con esattamente un acquisto
ALMENO_UNO = π_CF_Cliente(ACQUISTO)
ALMENO_DUE = π_CF_Cliente(A1 ⋈_(A1.CF_Cliente=A2.CF_Cliente AND
A1.Prodotto≠A2.Prodotto) A2)
ESATTAMENTE_UNO = ALMENO_UNO − ALMENO_DUE
```

3. Trovare elementi che soddisfano "tutti" (divisione relazionale)

```
// Studenti iscritti a tutti i corsi
π_Studente,Corso(ISCRIZIONE) ÷ π_Corso(CORSO)
```

4. Trovare massimi/minimi

```
// Persone che non hanno l'età massima (quindi trovare massimi)
P1 = PERSONA
P2 = PERSONA
π_P1.CF(P1) − π_P1.CF(P1 ⋈_(P1.Età<P2.Età) P2)
```

3.3 Esempi dagli esami

Esempio 1: Clienti che nel 2023 hanno prenotato in esattamente un unico stabilimento

```
P1 = σ_Data>=1/1/2023 AND Data<=31/12/2023(PRENOTAZIONE)
P2 = P1
P3CF = π_P1.CF-Cliente(P1 ⋈_P1.CF-Cliente=P2.CF-Cliente AND
P1.IdStabilimento≠P2.IdStabilimento P2)
π_CF-Cliente(P1) − P3CF
```

Esempio 2: Tipo di aereo con max passeggeri tra quelli usati per voli da Roma

```
C1 = σ_CittàPart='Roma'(VOLO ⋈ AEREO)
C2 = C1
TIPO-AEREO-NON-MASSIMO = π_C1.TipoAereo(C1
⋈_C1.NumPasseggeri<C2.NumPasseggeri C2)
(π_TipoAereo(C1)) \ TIPO-AEREO-NON-MASSIMO
```

Esempio 3: Codici fiscali delle persone che hanno visitato almeno due biblioteche

```
F1 = Frequentazione
F2 = Frequentazione
π_F1.CFPersona(F1 ⋈_F1.CFPersona=F2.CFPersona AND
F1.CodiceBiblio≠F2.CodiceBiblio F2)
```

Esempio 4: Le città in cui nessuna scuola ha avuto diplomati con 100 nel 2021

```
CITTA_CON_100_NEL_2021 = π_citta(σ_Voto=100 AND anno=2021(Diplomato ⋈
Scuola))
π_citta(SCUOLA) − CITTA_CON_100_NEL_2021
```

---

4. SQL

4.1 Elementi fondamentali

SELECT di base

```sql
SELECT [DISTINCT] colonna1, colonna2, ...
FROM tabella1 [alias1], tabella2 [alias2], ...
[WHERE condizione]
[GROUP BY colonna1, colonna2, ...]
[HAVING condizione_gruppo]
[ORDER BY colonna1 [ASC|DESC], colonna2 [ASC|DESC], ...]
[LIMIT n]
```

Operatori di confronto

- `=`, `<>` (o `!=`), `<`, `<=`, `>`, `>=`
- `BETWEEN a AND b` - Tra due valori (inclusi)
- `IN (valore1, valore2, ...)` - In un insieme di valori
- `LIKE pattern` - Corrispondenza con pattern (% = qualsiasi sequenza, _ = singolo carattere)
- `IS NULL`, `IS NOT NULL` - Confronto con NULL

Operatori logici

- `AND` - Entrambe le condizioni vere
- `OR` - Almeno una condizione vera
- `NOT` - Negazione

Funzioni di aggregazione

- `COUNT(*)` - Conta tutte le righe
- `COUNT(colonna)` - Conta valori non NULL
- `SUM(colonna)` - Somma
- `AVG(colonna)` - Media
- `MAX(colonna)` - Massimo
- `MIN(colonna)` - Minimo

4.2 JOIN

INNER JOIN

```
SELECT *
FROM tabella1
JOIN tabella2 ON tabella1.colonna = tabella2.colonna
```

LEFT JOIN

```
SELECT *
FROM tabella1
LEFT JOIN tabella2 ON tabella1.colonna = tabella2.colonna
```

RIGHT JOIN

```
SELECT *
FROM tabella1
RIGHT JOIN tabella2 ON tabella1.colonna = tabella2.colonna
```

FULL JOIN

```
SELECT *
FROM tabella1
FULL JOIN tabella2 ON tabella1.colonna = tabella2.colonna
```

4.3 Subquery e viste

Subquery nel WHERE

```
SELECT *
FROM tabella1
WHERE colonna IN (SELECT colonna FROM tabella2 WHERE condizione)
```

Subquery nel FROM

```
SELECT *
FROM (SELECT * FROM tabella WHERE condizione) AS subquery
```

Subquery nel SELECT

```
SELECT colonna1,
       (SELECT MAX(colonna) FROM tabella2) AS max_val
FROM tabella1
```

CREATE VIEW

```
CREATE VIEW nome_vista AS
SELECT colonna1, colonna2, ...
FROM tabella
WHERE condizione
```

4.4 Pattern comuni negli esami

1. Valore massimo/minimo

```
-- Città con più abitanti
SELECT città
FROM popolazione
WHERE abitanti = (SELECT MAX(abitanti) FROM popolazione)
```

2. Aggregazioni con GROUP BY

```
-- Media stipendi per dipartimento
SELECT dipartimento, AVG(stipendio) AS stipendio_medio
FROM impiegati
GROUP BY dipartimento
```

3. Condizioni sui gruppi (HAVING)

```
-- Dipartimenti con più di 10 impiegati
SELECT dipartimento, COUNT(*) AS num_impiegati
FROM impiegati
GROUP BY dipartimento
HAVING COUNT(*) > 10
```

4. Pattern "tutti" (quantificatore universale)

```sql
-- Studenti che hanno superato tutti gli esami
SELECT Studente
FROM Esami_Superati ES
GROUP BY Studente
HAVING COUNT(DISTINCT Corso) = (SELECT COUNT(*) FROM Corsi)
```

5. Pattern "nessuno" (NOT EXISTS, NOT IN)

```sql
-- Studenti senza voti sotto 25
SELECT DISTINCT Studente
FROM Esami
WHERE Studente NOT IN (
    SELECT Studente
    FROM Esami
    WHERE Voto < 25
)
```

4.5 Esempi dagli esami

Esempio 1: Stabilimento con più clienti in media giornalmente

```sql
CREATE VIEW CLIENTIXSTABILMENTOXDATA AS
SELECT IdStabilimento, Data, COUNT(*) as Numero
FROM PRENOTAZIONE
GROUP BY IdStabilimento, Data;

CREATE VIEW NUM_AVG_CLIENTIXSTABILMENTO AS
SELECT IdStabilimento, AVG(Numero) as MediaClienti
FROM CLIENTIXSTABILMENTOXDATA
GROUP BY IdStabilimento;

SELECT IdStabilimento
FROM NUM_AVG_CLIENTIXSTABILMENTO
WHERE MediaClienti = (
    SELECT MAX(MediaClienti)
    FROM NUM_AVG_CLIENTIXSTABILMENTO
);
```

Esempio 2: Città da cui partono almeno 100 voli settimanalmente

```sql
SELECT CittaPart, Count(*)/7
FROM VOLO
GROUP BY CittaPart
HAVING Count(*) >= 100;
```

Esempio 3: Giorno con più voli in arrivo a Roma

```sql
CREATE VIEW VOLI_PER_GIORNO_ROMA AS
SELECT GiornoSett, COUNT(*) as Num
FROM VOLO
WHERE CittàArr='Roma'
GROUP BY GiornoSett;

SELECT GiornoSett
FROM VOLI_PER_GIORNO_ROMA
WHERE Num = (SELECT MAX(Num) FROM VOLI_PER_GIORNO_ROMA);
```

Esempio 4: Squadre con goals in casa nella stagione 2023-2024

```sql
SELECT SQUADRACASA, SUM(GOALCASA) AS SOMMA_GOAL
FROM PARTITA
WHERE STAGIONE="2023-2024"
GROUP BY SQUADRACASA
ORDER BY SOMMA_GOAL DESC;
```

Esempio 5: Squadre con almeno 10 derby e goal medi

```sql
SELECT L1.SQUADRA, AVG(GOALCASA)
FROM PARTITA, LUOGO L1, LUOGO L2
WHERE L1.SQUADRA=SQUADRACASA
  AND L2.SQUADRA=SQUADRATRASFERTA
  AND L1.CITTA=L2.CITTA
GROUP BY L1.SQUADRA
HAVING COUNT(L1.SQUADRA) > 9;
```

---

5. Normalizzazione

5.1 Dipendenze funzionali

- **Definizione**: X → Y indica che i valori di X determinano univocamente i valori di Y
- **Proprietà**:
  - Riflessività: X → X
  - Aumentatività: se X → Y, allora XZ → Y
  - Transitività: se X → Y e Y → Z, allora X → Z

5.2 Chiavi di una relazione

- **Superchiave**: insieme di attributi che identificano univocamente una tupla
- **Chiave**: superchiave minimale (nessun sottoinsieme è una superchiave)
- **Chiave primaria**: chiave scelta per identificare le tuple

Algoritmo per trovare le chiavi

1. Calcolare la chiusura di ogni attributo e insieme di attributi
2. Se la chiusura include tutti gli attributi della relazione, è una superchiave
3. Verificare se è minimale (rimuovendo un attributo non è più una superchiave)

5.3 Forme normali

Prima Forma Normale (1NF)

- Ogni attributo contiene valori atomici (non divisibili)
- Non ci sono gruppi ripetuti

Seconda Forma Normale (2NF)

- È in 1NF
- Ogni attributo non-chiave dipende dall'intera chiave

Terza Forma Normale (3NF)

- È in 2NF
- Ogni attributo non-chiave dipende solo dalla chiave e non da altri attributi non-chiave

Forma Normale di Boyce-Codd (BCNF)

- Per ogni dipendenza funzionale $X \rightarrow Y$, X è una superchiave

5.4 Copertura ridotta di dipendenze funzionali

Algoritmo per la copertura ridotta

1. **Scomposizione del lato destro**: trasformare $X \rightarrow YZ$ in $X \rightarrow Y$ e $X \rightarrow Z$
2. **Eliminazione degli attributi ridondanti a sinistra**: rimuovere attributi non necessari
3. **Eliminazione di dipendenze ridondanti**: rimuovere dipendenze derivabili da altre

5.5 Decomposizione in 3NF

Algoritmo di decomposizione in 3NF

1. Trovare una copertura ridotta G delle dipendenze funzionali

2. Per ogni gruppo di dipendenze X → Y in G con la stessa parte sinistra X, creare una relazione R(X,Y)
3. Se nessuna relazione contiene una chiave della relazione originale, aggiungere una relazione che contiene una chiave
4. Eliminare relazioni contenute in altre

5.6 Esempi di normalizzazione

Esempio 1: R(A,B,C,D,E) con dipendenze {B→C, B→E, C→A, C→D}

1. **Trovare le chiavi**:
   - B+ = {B,C,E,A,D} (include tutti gli attributi)
   - C+ = {C,A,D} (manca B,E)
   - Chiavi: B
2. **Verificare violazioni 3NF/BCNF**:
   - C→A viola BCNF perché C non è superchiave
   - C→D viola BCNF perché C non è superchiave
3. **Decomposizione in 3NF**:
   - R1(B,C,E)
   - R2(C,A,D)
4. **Verifica se la decomposizione è in BCNF**:
   - In R1: B→C, B→E rispettano BCNF perché B è chiave di R1
   - In R2: C→A, C→D rispettano BCNF perché C è chiave di R2
5. **Verifica della perdita di join**:
   - Attributo comune: C
   - C è chiave di R2
   - La decomposizione è senza perdita

Esempio 2: R(A,B,C,D,E) con dipendenze {AB→CDE, B→C, C→B, D→E}

1. **Trovare le chiavi**:
   - AB+ = {A,B,C,D,E}
   - B+ = {B,C}
   - C+ = {C,B}
   - D+ = {D,E}
   - Chiavi: AD, AC
2. **Copertura ridotta**:
   - B→C
   - C→B

- D→E
- A,B→D (oppure A,C→D)

3. **Decomposizione in 3NF**:
   - R1(B,C)
   - R2(D,E)
   - R3(A,B,D) o R3(A,C,D)
   - Aggiungere R4(A,D) che contiene una chiave

---

6. Transazioni e Ripristino

6.1 Concetti di base delle transazioni

- **Transazione**: sequenza di operazioni di lettura e scrittura trattate come unità atomica
- **Proprietà ACID**:
  - **A**tomicità: eseguita completamente o per nulla
  - **C**oerenza: mantiene la base dati in uno stato coerente
  - **I**solamento: gli effetti non sono visibili ad altre transazioni prima del commit
  - **D**urabilità: gli effetti persistono anche dopo un guasto

6.2 Schedules e serializzabilità

Tipi di schedule

- **Seriale**: le transazioni vengono eseguite sequenzialmente
- **Non seriale**: le operazioni di transazioni diverse sono intercalate

Conflict-serializzabilità

- Due operazioni sono in conflitto se:
  - Appartengono a transazioni diverse
  - Accedono allo stesso elemento
  - Almeno una è un'operazione di scrittura
- **Test di conflict-serializzabilità**:
  1. Costruire il grafo dei conflitti (nodi = transazioni, archi = conflitti)
  2. Lo schedule è conflict-serializzabile se e solo se il grafo è aciclico

View-serializzabilità

- Due schedule sono view-equivalenti se:
  - Preservano la relazione "leggi-da" (ogni transazione legge gli stessi valori)
  - Hanno le stesse scritture finali

6.3 Ripristino del database

Tipi di guasti

- **Guasto di sistema** (soft crash): perdita di memoria principale
- **Guasto di dispositivo** (hard crash): perdita di memoria secondaria

Log delle transazioni

- Registra tutte le operazioni su database
- **Formato**: <T, X, old_val, new_val> per update
- **Checkpoint**: punto di sincronizzazione tra log e DB

Ripristino a freddo (dopo guasto di dispositivo)

1. Ripristinare il DB dall'ultimo backup
2. Applicare tutte le transazioni dal momento del backup

Ripristino a caldo (dopo guasto di sistema)

1. **Analisi**: identificare transazioni da completare (REDO) e da annullare (UNDO)
2. **REDO**: rieseguire transazioni committed
3. **UNDO**: annullare transazioni non committed

6.4 Algoritmo di ripristino immediato

Operazioni nel log

- **B(T)**: Begin transaction T
- **C(T)**: Commit transaction T
- **A(T)**: Abort transaction T
- **U(T,X,old,new)**: Update di X da old a new nella transazione T
- **I(T,X,new)**: Insert di X con valore new nella transazione T
- **D(T,X,old)**: Delete di X con valore old nella transazione T
- **CK(T1,T2,...)**: Checkpoint delle transazioni T1, T2, ...

Algoritmo per il ripristino

1. **Determinare transazioni da REDO e UNDO**:
   - REDO = {T | log contiene C(T) dopo l'ultimo checkpoint}
   - UNDO = {T | log contiene B(T) ma non C(T) o A(T) dopo l'ultimo checkpoint}
2. **Eseguire UNDO** (dalla fine del log verso l'inizio):
   - Per ogni scrittura W(T,X,old,new) con T in UNDO: X=old
3. **Eseguire REDO** (dall'inizio del log verso la fine):

- Per ogni scrittura W(T,X,old,new) con T in REDO: X=new

6.5 Esempi di ripristino

Esempio 1: Ripristino a caldo

```
Log: B(T1), U(T1,X,10,20), B(T2), U(T2,Y,30,40), C(T2), U(T1,Z,50,60),
guasto
```

1. **Analisi**:
   - REDO = {T2} (transazioni committed)
   - UNDO = {T1} (transazioni non committed)
2. **UNDO**:
   - Z = 50 (annulla U(T1,Z,50,60))
   - X = 10 (annulla U(T1,X,10,20))
3. **REDO**:
   - Y = 40 (riapplica U(T2,Y,30,40))

Esempio 2: Ripristino dopo guasto di dispositivo

```
Log: DUMP, B(T5), B(T6), CK(T5,T6), B(T7), U(T7,O6,B6,A6), U(T6,O3,B7,A7),
B(T8), I(T8,O5,A5), C(T8), A(T5)
```

1. **Ripristino a freddo**:
   - Ripristinare il DB dall'ultimo DUMP
   - Rieseguire tutte le operazioni dopo DUMP
2. **Ripristino a caldo**:
   - REDO = {T8} (committed)
   - UNDO = {T5, T6, T7} (non committed)
3. **UNDO**:
   - O3 = B7 (annulla U(T6,O3,B7,A7))
   - O6 = B6 (annulla U(T7,O6,B6,A6))
4. **REDO**:
   - O5 = A5 (riapplica I(T8,O5,A5))

---

7. Indici e Ottimizzazione

7.1 Indici

Tipi di indici

- **B-Tree**: struttura ad albero bilanciato, efficiente per range query
- **B+Tree**: variante del B-Tree con valori solo nelle foglie, ottimizzato per range query
- **Hash**: struttura basata su funzione hash, efficiente per ricerche puntuali

Quando usare i diversi tipi di indici

Operazione	B-Tree	B+Tree	Hash
Ricerca puntuale (=)	Buono	Buono	Ottimo
Range query (<, >, BETWEEN)	Buono	Ottimo	Non supportato
ORDER BY	Buono	Ottimo	Non supportato
JOIN	Buono	Buono	Limitato

7.2 Scelta degli indici

Criteri di scelta

1. **Selettività**: quanto è selettivo l'attributo?
2. **Frequenza di query**: quanto spesso viene usato nelle query?
3. **Costo di manutenzione**: quanto costa mantenere l'indice durante insert/update/delete?

Indici su più attributi

- **Ordine degli attributi**: più selettivo → meno selettivo
- **Indice composito vs indici multipli**: dipende dalle query tipiche

7.3 Ottimizzazione delle query

Tecniche di ottimizzazione

1. **Limite della proiezione**: selezionare solo gli attributi necessari
2. **Limite della selezione**: filtrare il prima possibile
3. **Ordine dei join**: iniziare dai join più selettivi
4. **Uso di indici**: scegliere gli indici appropriati

Esempi di ottimizzazione

1. **Query originale**:

```
SELECT *
FROM Cliente c, Ordine o, Prodotto p
```

```
WHERE c.id = o.cliente_id AND o.prodotto_id = p.id
```

2. **Query ottimizzata**:

```
SELECT c.nome, o.data, p.descrizione
FROM Ordine o
JOIN Cliente c ON o.cliente_id = c.id
JOIN Prodotto p ON o.prodotto_id = p.id
```

7.4 Quiz tipici sugli indici

Esempio 1: Selezione dell'indice ottimale

Per la query:

```
SELECT * FROM R WHERE B='Valore'
```

L'indice migliore è:

- Un indice Hash su B

Esempio 2: Selezione dell'indice per range query

Per la query:

```
SELECT * FROM R WHERE C BETWEEN 10 AND 20
```

L'indice migliore è:

- Un indice B+Tree su C

Esempio 3: Selezione dell'indice per aggregazione

Per la query:

```
SELECT MIN(A) FROM R WHERE B=10
```

L'indice migliore è:

- Un indice B+Tree sulla coppia (B,A)

---

8. Quiz Tipici

8.1 Domande sulle relazioni

Cardinalità del risultato

Domanda: Data la relazione R(A,B,C) e S(D,E,F), quale affermazione è vera per $R \bowtie_{A=D} S$?

- Risposta: $0 \leq |R \bowtie_{A=D} S| \leq |R| \times |S|$

Proprietà delle chiavi

Domanda: Data R(A,B,C) con dipendenze A→BC e B→A, quale affermazione è vera?

- Risposta: AB è superchiave ma non è chiave (perché B è già chiave)

8.2 Domande sulle forme normali

Domanda: Data R(A,B,C,D) con dipendenze AB→C e C→D, in quale forma normale è?

- Risposta: È in 3NF ma non in BCNF (C→D viola la BCNF perché C non è superchiave)

8.3 Domande sugli indici

Domanda: Per la query SELECT * FROM S WHERE X=4 AND Y>7, quale indice è più efficiente?

- Risposta: Indice B+Tree sulla coppia (X,Y)

8.4 Domande sulle transazioni

Domanda: Dato lo schedule S=r1(x)w2(x)r3(x)w1(u)w3(v)r3(y)r2(y)w3(u)w4(t)w3(t), è conflict-serializzabile?

- Risposta: Sì, perché il grafo dei conflitti è aciclico.

8.5 Domande sul ripristino

Domanda: Dato il log CK(T5,T6), B(T7), U(T7,O6,B6,A6), U(T6,O3,B7,A7), B(T8), C(T7), I(T8,O5,A5), quali transazioni richiedono REDO?

- Risposta: T7, T8 (transazioni committed dopo l'ultimo checkpoint)