

Appunti sulla Progettazione Logica dei Database

Basato sulle slide del Prof. Danilo Montesi

16 maggio 2025

Indice

1	Introduzione alla Progettazione Logica dei Database	2
1.1	Input della Progettazione Logica	2
1.2	Output della Progettazione Logica	2
1.3	Non è una semplice traduzione!	2
2	Fasi della Trasformazione Logica	2
3	Analisi delle Prestazioni (Approssimata)	2
4	Attività di Ristrutturazione dello Schema E-R	3
4.1	Analisi delle Ridondanze	3
4.2	Eliminazione delle Generalizzazioni (Gerarchie)	4
4.3	Partizionamento/Raggruppamento di Entità e Relazioni	4
4.4	Identificazione delle Chiavi Primarie	5
5	Traduzione nel Modello Relazionale (Regole Generali)	5
6	Attenzione Finale	6
7	Strumenti (Tools)	6

1 Introduzione alla Progettazione Logica dei Database

L'obiettivo principale della **progettazione logica** è "tradurre" lo schema concettuale (spesso un diagramma E-R) in uno schema logico (ad esempio, per un database relazionale come SQL Server, MySQL, PostgreSQL) che rappresenti gli stessi dati in modo **corretto ed efficiente**.

1.1 Input della Progettazione Logica

- **Schema Concettuale:** Il diagramma E-R che descrive la realtà di interesse.
- **Informazioni sul Carico Applicativo (Workload):** Quali operazioni verranno eseguite più frequentemente? Quanti dati ci aspettiamo? (Fondamentale per l'efficienza).
- **Modello Logico Scelto:** Ad es. relazionale, a oggetti, a grafo. Ci concentreremo sul relazionale.

1.2 Output della Progettazione Logica

- **Schema Logico:** Ad esempio, un insieme di istruzioni `CREATE TABLE` per SQL.
- **Documentazione Associata:** Spiegazioni delle scelte fatte.

1.3 Non è una semplice traduzione!

- Alcuni aspetti dello schema concettuale potrebbero non essere rappresentabili direttamente nel modello logico scelto (es. generalizzazioni nel modello relazionale puro).
- Bisogna considerare le **prestazioni (efficienza)**.

2 Fasi della Trasformazione Logica

1. Ristrutturazione dello Schema E-R (E-R Schema Restructuring):

- Si modifica lo schema E-R iniziale tenendo conto del carico applicativo e del modello logico.
- **Perché?**
 - Semplificare la successiva traduzione.
 - Ottimizzare le prestazioni.
- **Nota Bene:** Uno schema E-R ristrutturato *non è più* uno schema concettuale puro, perché inizia a includere considerazioni di implementazione.

2. Traduzione nel Modello Logico:

- Si converte lo schema E-R ristrutturato nello schema del modello scelto (es. tabelle relazionali).

3 Analisi delle Prestazioni (Approssimata)

A livello concettuale/logico iniziale, non possiamo valutare le prestazioni con precisione assoluta, ma usiamo degli "indicatori":

- **Spazio (Space):** Numero di istanze (righe nelle tabelle) attese.
 - *Esempio Pratico:* Se hai una tabella `Utenti` e prevedi 1 milione di utenti, questo è un indicatore di spazio.
- **Tempo (Time):** Numero di istanze (entità e relazioni) "visitare" (lette/scritte) durante un'operazione.
 - *Esempio Pratico:* Per mostrare il profilo di un utente con i suoi ultimi 10 post e i commenti a quei post, quante righe da diverse tabelle (`Utenti`, `Post`, `Commenti`) devi leggere?

La **Tabella delle Dimensioni (Size Table)** stima il numero di istanze per ogni entità (E) e relazione (R). La **Tabella degli Accessi (Access Table)**, per un'operazione specifica, elenca:

- Quali entità/relazioni vengono accedute.
- Quante volte (numero accessi).
- Tipo di accesso (Lettura/Scrittura).
- Ordine di accesso.

Questo aiuta a confrontare diverse opzioni di ristrutturazione.

4 Attività di Ristrutturazione dello Schema E-R

Sono 4 attività principali:

4.1 Analisi delle Ridondanze

- Una **ridondanza** è un'informazione rilevante che può essere derivata da altre informazioni già presenti.
- Bisogna decidere se: mantenere, rimuovere o *creare nuove* ridondanze.
- **Pro della Ridondanza:**
 - Semplifica le query (meno join, calcoli al volo).
 - *Esempio Pratico:* In una tabella `Ordini`, potresti avere una colonna `TotaleOrdine`. Questo è ridondante se puoi calcolarlo sommando `Prezzo * Quantità` da una tabella `RigheOrdine` collegata. Averlo precalcolato velocizza la lettura del totale ordine.
- **Contro della Ridondanza:**
 - Gli aggiornamenti richiedono più tempo (devi aggiornare il dato in più posti e mantenerlo consistente).
 - Aumenta lo spazio di archiviazione.
 - *Esempio Pratico (continuazione):* Se modifichi una riga in `RigheOrdine`, devi ricalcolare e aggiornare `TotaleOrdine` nella tabella `Ordini`. Se non lo fai, i dati diventano inconsistenti.
- **Tipi di Ridondanze comuni in E-R:**
 - **Attributi derivabili:**
 - * Da altri attributi nella stessa entità/relazione (es. `Fattura` con `ValoreNetto`, `IVA`, `ValoreLordo`. `ValoreLordo` è derivabile).
 - * Da attributi di altre entità/relazioni (es. `Acquisto` con attributo `Totale`, derivabile da $\sum(\text{Composizione}.\text{Quantità} \cdot \text{Prodotto}.\text{Prezzo})$).
 - **Relazioni derivabili:** Spesso cicli di relazioni (es. se hai `Studente` - `Frequenta` - `Lezione` - `TenutaDa` - `Docente`, una relazione diretta `Studente` - `InsegnatoDa` - `Docente` potrebbe essere ridondante).
 - **Attributi derivabili da conteggio:** (es. `Città` con attributo `NumeroAbitanti`, derivabile da `COUNT(*)` delle persone che risiedono in quella città).
- **Decisione sulla Ridondanza:** Si basa sull'analisi costi/benefici, considerando la frequenza delle operazioni.
 - Se un dato derivato è letto molto frequentemente e scritto/aggiornato raramente, mantenerlo ridondante può essere vantaggioso.
 - Se è aggiornato spesso, la ridondanza può diventare problematica.

4.2 Eliminazione delle Generalizzazioni (Gerarchie)

- Il modello relazionale puro non supporta direttamente le generalizzazioni (ereditarietà). Vanno trasformate.
- **Tre Soluzioni Possibili** (esempio: E0 genitore, E1 ed E2 figli):
 1. **Embedding dei Figli nel Genitore (Roll-up / Accorpamento verso l'alto):**
 - Si elimina E1 ed E2. L'entità E0 eredita tutti gli attributi di E1 ed E2.
 - Si aggiunge un attributo "Tipo" (o "Kind") a E0 per distinguere se un'istanza era originariamente E1 o E2.
 - Gli attributi specifici di E1 o E2 diventano NULLabili in E0 se un'istanza non è di quel tipo.
 - Le relazioni dei figli vengono "spostate" sul genitore.
 - *Esempio Pratico:* Veicolo (genitore), Auto (figlio), Moto (figlio). Diventa un'unica tabella VEICOLI(targa, marca, modello, tipoVeicolo, numPorte_auto, cilindrata_moto, ...). numPorte_auto sarà NULL per le moto.
 - **Quando usarla:** Se le operazioni accedono frequentemente al genitore e ai figli contemporaneamente.
 2. **Embedding del Genitore nei Figli (Roll-down / Accorpamento verso il basso):**
 - Si elimina E0. Le entità E1 ed E2 ereditano tutti gli attributi di E0.
 - Le relazioni di E0 vengono replicate per E1 ed E2.
 - *Esempio Pratico:* Tabelle separate AUTO(targa, marca, modello_veicolo, numPorte, ...) e MOTO(targa, marca, modello_veicolo, cilindrata, ...). marca e modello_veicolo sono duplicati.
 - **Quando usarla:** Se le operazioni accedono ai figli indipendentemente l'uno dall'altro.
 3. **Sostituzione della Generalizzazione con Relazioni (Una tabella per classe):**
 - Si mantengono E0, E1, E2 come entità separate.
 - Si creano relazioni 1-a-1 tra E0 ed E1, e tra E0 ed E2. La chiave primaria di E1 (e E2) sarà anche chiave esterna verso E0.
 - *Esempio Pratico:* Tabella VEICOLI(targa_PK, marca, modello). Tabella AUTO(targa_FK_PK, numPorte). Tabella MOTO(targa_FK_PK, cilindrata). Per avere tutti i dati di un'auto, fai un JOIN.
 - **Quando usarla:** Se i figli sono acceduti indipendentemente dal padre.
- **Soluzioni Ibride:** Si possono combinare queste strategie, specialmente con gerarchie a più livelli.

4.3 Partizionamento/Raggruppamento di Entità e Relazioni

L'obiettivo è ridurre il numero di accessi.

- **Partizionamento Verticale di Entità:**
 - Se un'entità ha molti attributi e alcuni sono usati frequentemente insieme, mentre altri raramente, si può dividere l'entità in due (o più) entità collegate da una relazione 1-a-1.
 - *Esempio Pratico:* Impiegato(Codice, Cognome, Indirizzo, DataNascita, Livello, Salario, Tasse) può diventare DatiPersonali(Codice_PK, Cognome, Indirizzo, DataNascita) e DatiAziendali(Codice_PK_FK, Livello, Salario, Tasse).
- **Ristrutturazione di Attributi Multi-Valore:**
 - Un attributo multi-valore (es. Ufficio con Telefono(1,N)) viene trasformato in una nuova entità (Telefono) e una relazione 1-a-N (Possiede).
 - *Esempio Pratico:* Se un Prodotto può avere più Tag, crei una tabella Prodotti, una tabella Tag e una tabella di join ProdottoTag.
- **Raggruppamento di Entità (Denormalizzazione):**

- Se un'entità A ha una relazione 1-a-1 (o N-a-1) con un'entità B, e sono *sempre* accedute insieme, gli attributi di B possono essere "assorbiti" in A.
- *Esempio Pratico*: Persona(0,1) -- ViveIn -- (1,1)Appartamento. Si possono spostare NumAppartamento e IndirizzoAppartamento nell'entità Persona, rendendoli NULLabili.

- **Partizionamento Orizzontale di Relazioni:**

- Si dividono le istanze di una relazione in più relazioni distinte se hanno pattern di accesso diversi.
- *Esempio Pratico*: Relazione ParteDi tra Giocatore e Squadra può essere divisa in MilitaAttualmenteIn e HaMilitatoInPassato.

4.4 Identificazione delle Chiavi Primarie

- Operazione **obbligatoria** per la traduzione in un modello relazionale.
- **Criteri di Scelta:**
 1. **Informazione Obbligatoria:** L'attributo deve essere NOT NULL.
 2. **Semplicità:** Preferire chiavi singole a chiavi composte.
 3. **Usata nelle Operazioni più Frequenti/Rilevanti.**
- **Nuovi Attributi (Surrogate Keys):** Se nessuna combinazione di attributi esistenti è una buona chiave primaria, si introducono nuovi attributi "artificiali".
 - *Esempio Pratico*: id INT AUTO_INCREMENT PRIMARY KEY in SQL, ObjectId() in MongoDB.

5 Traduzione nel Modello Relazionale (Regole Generali)

- **Entità:** Diventano tabelle. Gli attributi dell'entità diventano colonne. La chiave primaria identificata diventa la PRIMARY KEY.
- **Relazioni:**

- **Molti-a-Molti (M:N):**

- * Diventano una **nuova tabella** (tabella di associazione).
- * Colonne: chiavi primarie delle entità coinvolte (come FOREIGN KEY, formano la PRIMARY KEY della nuova tabella) + attributi propri della relazione.
- * *Esempio*: IMPIEGATO(0,N) -- ISCRIZIONE(DataInizio) -- (1,N)PROGETTO

```
IMPIEGATO(Numero_PK, Cognome, Salario)
PROGETTO(Codice_PK, Nome, Budget)
ISCRIZIONE(NumeroImpiegato_FK_PK, CodiceProgetto_FK_PK, DataInizio)
```

- * **Vincoli di Integrità Referenziale:** Vanno definiti.
- * **Nomi Espressivi per FK:** Meglio IDImpiegato, IDProgetto.
- * **Cardinalità Minima:** La traduzione base M:N *non* cattura la cardinalità minima. Richiede logica applicativa o TRIGGER/CHECK complessi.

- **Relazioni Ricorsive (M:N sulla stessa entità):**

- * Simile a M:N, si crea una nuova tabella con due chiavi esterne che puntano alla stessa tabella originale.
- * *Esempio*: PRODOTTO -- CompostoDa(NumeroPezzi) -- PRODOTTO

```
PRODOTTO(Codice_PK, Nome, Costo)
COMPOSTODA(CodiceProdottoComposto_FK_PK, CodiceComponente_FK_PK, NumeroPezzi)
```

- **Relazioni N-arie (coinvolgono 3 o più entità):**

- * Diventano una nuova tabella con le chiavi primarie di tutte le entità coinvolte (come FK) + attributi propri.

* *Esempio:* FORNITORE -- FORNITURA(NumeroPezzi) -- PRODOTTO -- A REPARTO

```
FORNITURA(ID_Fornitore_FK_PK, ID_Prodotto_FK_PK, ID_Reparto_FK_PK, NumeroPezzi)
```

– Uno-a-Molti (1:N):

- * La chiave primaria dell'entità sul lato "1" viene **propagata** come FOREIGN KEY nell'entità sul lato "N".
- * Gli attributi della relazione vengono aggiunti alla tabella sul lato "N".
- * *Esempio:* GIOCATORE(1,1) -- CONTRATTO(DataIngaggio) -- (0,N)SQUADRA

```
SQUADRA(Nome_PK, Citta, ColoriSociali)
GIOCATORE(CF_PK, Cognome, DataNascita, Ruolo, NomeSquadra_FK, DataIngaggio)
```

- * **Cardinalità Minima (0 sul lato N):** Se opzionale, la FOREIGN KEY permette NULL.
- * **Cardinalità Minima (1 sul lato N):** Se obbligatoria, la FOREIGN KEY è NOT NULL.

– Entità con Identificatore Esterno (Entità Debole):

- * L'entità debole diventa una tabella la cui chiave primaria è composta dalla PK dell'entità forte + identificatore parziale.
- * *Esempio:* STUDENTE(Matricola) identificato da UNIVERSITA(Nome).

```
UNIVERSITA(Nome_PK, Citta, Indirizzo)
STUDENTE(NomeUniversita_FK_PKpart, Matricola_PKpart, Cognome, AnnoCorso)
```

La PK di STUDENTE è (NomeUniversita, Matricola).

– Uno-a-Uno (1:1):

- * Si sceglie una delle due tabelle e si aggiunge la PK dell'altra come FOREIGN KEY e UNIQUE KEY. Gli attributi della relazione vanno nella tabella scelta.
- * *Esempio:* CAPO(1,1) -- SUPERVISIONE(DataInizio) -- (1,1)DIPARTIMENTO

```
-- Opzione 1:
DIPARTIMENTO(Nome_PK, Ufficio, Telefono)
CAPO(Codice_PK, Cognome, Salario, NomeDipartimento_FK_UNIQUE,
↪ DataInizioSupervisione)

-- Opzione 2:
CAPO(Codice_PK, Cognome, Salario)
DIPARTIMENTO(Nome_PK, Ufficio, Telefono, CodiceCapo_FK_UNIQUE,
↪ DataInizioSupervisione)
```

* Cardinalità (0,1) - Opzionale:

- Se una partecipazione è opzionale, la FOREIGN KEY è NULLabile (ma sempre UNIQUE se presente).
- Se entrambe sono (0,1): la soluzione più pulita è una tabella separata per la relazione SUPERVISIONE(CodiceCapo_FK_PK, NomeDipartimento_FK_PK, DataInizio).

6 Attenzione Finale

Piccole differenze nelle cardinalità e nelle scelte degli identificatori nello schema E-R possono portare a significati e schemi logici molto diversi. È fondamentale essere precisi.

7 Strumenti (Tools)

Esistono software CASE (Computer-Aided Software Engineering) che supportano la modellazione, come:

- ERwin/ERX
- IBM Rational Rose

Questi strumenti aiutano a disegnare diagrammi E-R e a generare lo schema DDL.