Progettazione di una base di dati

Il processo di sviluppo di una base di dati si articola in diverse fasi, che possono essere sintetizzate così:

1. **Raccolta e analisi dei requisiti**: Si identificano le informazioni e le operazioni necessarie, collaborando con clienti e utenti per comprendere l'universo di riferimento. Si individuano le entità principali e le relazioni tra di esse.

2. **Progettazione concettuale**: Si crea uno schema concettuale indipendente dal tipo di database, spesso usando il modello Entità-Relazione (ER) o l'UML. Questo schema rappresenta dati, attributi e relazioni, ma non i dettagli delle transazioni.

3. **Scelta del tipo di database**: Si decide quale modello di dati adottare (ad esempio, relazionale, a oggetti, documentale) prima di procedere con la progettazione logica.

4. **Progettazione logica**: Lo schema concettuale viene tradotto in uno schema logico adatto al modello di dati scelto. Ad esempio, nel modello relazionale, le entità e le relazioni diventano tabelle con attributi specifici.

5. **Scelta del DBMS**: Si seleziona il software di gestione del database più adatto (es. MySQL, PostgreSQL), considerando aspetti tecnici, economici e organizzativi.

6. **Progettazione fisica**: Lo schema logico viene implementato concretamente nel linguaggio del DBMS scelto (es. SQL). Si definiscono tabelle, vincoli, viste e strutture di memorizzazione per ottimizzare le prestazioni.

7. **Realizzazione e ottimizzazione**: Si creano le tabelle, si caricano i dati e si testano le transazioni. Durante questa fase si monitora il sistema per migliorare l'efficienza (tuning).

Il processo è teoricamente lineare, ma spesso include cicli di revisione. Gli errori nelle fasi iniziali hanno un impatto maggiore rispetto a quelli nelle fasi finali, che sono più facili da correggere.

I linguaggi per accedere alle basi di dati si suddividono principalmente in due categorie:

1. **Linguaggio di definizione dei dati (DDL)**

- Consente di definire i tre livelli di astrazione del database: esterno (viste), logico (struttura dati) e fisico (memorizzazione).
- Esempio: creare tabelle, definire vincoli, o configurare viste.

2. **Linguaggio di manipolazione dei dati (DML)**

- Permette di gestire i dati: inserire, modificare, eliminare e recuperare informazioni.
- Può essere:
    - **Procedurale**: specifica *cosa* fare e *come* farlo.
    - **Dichiarativo**: specifica solo *cosa* ottenere; il sistema decide il metodo di esecuzione.
- SQL è il linguaggio dichiarativo più usato per definire e manipolare dati nei database relazionali.
    - Una **query** è un'istruzione SQL per interrogare i dati.

Accesso ai database

L'accesso diretto con SQL è tipico di amministratori, progettisti e utenti esperti. Tuttavia, la maggior parte degli utenti interagisce con i database tramite applicazioni sviluppate in linguaggi come C, C++ o Java, che incorporano istruzioni SQL. Esistono due principali metodi di integrazione tra linguaggi ad alto livello e SQL:

1. **SQL Embedded**
    - Le istruzioni SQL vengono inserite direttamente nel codice sorgente del programma.
    - Prima della compilazione, un precompilatore traduce le istruzioni SQL in codice del linguaggio ospite.
2. **Call Level Interface (CLI)**
    - Si utilizza una libreria di funzioni o procedure nel linguaggio ospite per comunicare con il database.
    - Esempi di interfacce:
        - **ODBC (Open Database Connectivity)**: standard originariamente sviluppato da Microsoft.
        - **JDBC (Java Database Connectivity)**: progettato da Sun Microsystems per connettere Java ai database relazionali.

La progettazione di basi di dati relazionali si divide in due attività principali:

1. **Progettazione dei dati**: si definiscono struttura e organizzazione dei dati.
2. **Progettazione delle transazioni**: si stabiliscono le caratteristiche delle operazioni sui dati.

Queste due attività sono strettamente collegate e dovrebbero procedere parallelamente.

Fasi della progettazione di basi di dati

La metodologia standard si articola in quattro fasi principali:

1. **Raccolta e analisi dei requisiti**:
    - Si identificano i dati e le operazioni necessarie, lavorando a stretto contatto con i futuri utenti del sistema.
    - Risultato: una descrizione chiara e dettagliata di cosa rappresentare.
2. **Progettazione concettuale**:
    - Si realizza uno schema concettuale, indipendente dal tipo di database, che descrive le entità, i loro attributi e le relazioni.
    - Strumenti comuni: modello Entità-Relazione (ER).
3. **Progettazione logica**:
    - Lo schema concettuale viene trasformato in uno schema logico adatto al modello relazionale.
    - Si definiscono tabelle, colonne, chiavi e vincoli, senza ancora legarsi al DBMS specifico.

4. **Progettazione fisica**:
    - Lo schema logico viene tradotto in uno schema fisico implementabile su uno specifico DBMS.
    - Si definiscono strutture di memorizzazione, indici e ottimizzazioni per le prestazioni.

Realizzazione del database

Dopo la progettazione, si passa alla realizzazione, che prevede:

- Creazione fisica delle strutture dei dati (es. tabelle, viste).
- Implementazione delle applicazioni che utilizzeranno il database.

Principio fondamentale

La metodologia si basa sulla separazione tra:

- **Cosa rappresentare** (fasi di analisi e progettazione concettuale).
- **Come rappresentarlo** (fasi di progettazione logica e fisica).

Raccolta e Analisi dei Requisiti

La **raccolta dei requisiti** serve a identificare:

- **Caratteristiche statiche**: struttura e organizzazione dei dati.
- **Caratteristiche dinamiche**: operazioni o transazioni previste.

I requisiti, spesso provenienti da utenti, documentazione esistente o sistemi preesistenti, vengono raccolti in linguaggio naturale, ma possono risultare ambigui o disorganizzati. L'**analisi dei requisiti** ha lo scopo di chiarirli e organizzarli.

Metodi e Best Practice

- **Coinvolgimento del cliente**: Aumenta la soddisfazione e facilita la raccolta dei requisiti. Progettisti e utenti collaborano per definire il flusso di lavoro o tramite prototipi.
- **Progettazione contestuale**: I progettisti lavorano direttamente nell'ambiente operativo dell'applicazione.
- **Regole per una specifica chiara**:
    1. Usare un livello di astrazione uniforme.
    2. Evitare sinonimi e omonimi; usare sempre gli stessi termini.
    3. Preferire frasi brevi e semplici.
    4. Strutturare il testo in paragrafi dedicati a singole entità.

Componenti del Modello

1. **Entità**: Classi di oggetti autonomi e rilevanti.
    - Esempi: *teatro*, *dipendente*, *spettacolo*.
2. **Attributi**: Caratteristiche semplici associate alle entità.

    - Esempi: *nome del teatro*, *codice fiscale del dipendente*, *titolo dello spettacolo*.
3. **Relazioni**: Associazioni tra due o più entità, con specificazione delle cardinalità.
    - Esempio: un *teatro* ha più *dipendenti*, un *dipendente* può lavorare in più teatri.

Specifiche delle Transazioni

Accanto ai dati, si raccolgono specifiche sulle **transazioni tipiche**, cioè le operazioni più frequenti.

- **Regola 80/20**: L'80% del carico deriva dal 20% delle transazioni.
- **Ottimizzazione per efficienza**: Ad esempio:
    - Se una transazione richiede il numero di posti liberi per uno spettacolo, si registrano capienza e posti prenotati.
    - Se si richiedono spettacoli ordinati cronologicamente, si crea un indice sulla data.

Importanza della Progettazione Parallela

La progettazione delle **transazioni** è strettamente legata a quella dei **dati** e dovrebbe procedere in parallelo. Tuttavia, queste attività sono spesso separate:

- **Progettisti di basi di dati** modellano i dati.
- **Ingegneri del software** si occupano delle transazioni.

Una stretta collaborazione tra le due figure è cruciale per garantire un sistema efficiente e coerente.

Ecco una sintesi del **caso di studio** e i punti chiave da considerare nella progettazione della base di dati e delle transazioni per la rete teatrale.

Requisiti per i Dati

Entità e Attributi

1. **Teatro**
    - *Attributi*: nome, indirizzo, telefono, fax, email, pagina web.
2. **Spazio Teatrale**
    - *Attributi*: nome, indirizzo, pianta (grafico), capienza.
    - *Relazione*: può essere condiviso da più teatri.
3. **Biglietteria**
    - *Attributi*: nome, indirizzo, telefono, email, orari di apertura settimanali.
    - *Relazione*: associata a un unico teatro.
4. **Dipendente**
    - *Attributi*: codice fiscale, nome, cognome, dati personali, data di assunzione, ruolo, stipendio (storico e attuale).

- *Relazione*: un dipendente può lavorare in più teatri.
- *Regola*: con almeno 10 anni di servizio in un teatro, può accedere al CdA.

5. **Stagione Teatrale**

   - *Attributi*: nome, biennio.
   - *Relazione*: associata a uno o più teatri e composta da più spettacoli.

6. **Spettacolo**

   - *Attributi*: titolo, anno di produzione, descrizione, interpreti, produttori.
   - *Relazioni*:
     - Può essere parte di più stagioni di diversi teatri.
     - *Regola*: un teatro può produrre al massimo 2 spettacoli per stagione.

7. **Interprete**

   - *Attributi*: nome, ruolo, CV.

8. **Produttore**

   - *Attributi per enti esterni*: nome, indirizzo, email, telefono.
   - *Relazione*: può essere un teatro o un ente esterno.

9. **Messa in Scena**

   - *Attributi*: data, ora, spazio teatrale, immagini/video, prezzi dei biglietti (intero, ridotto, studenti), posti disponibili.
   - *Regola*: il prezzo ridotto è l'80% dell'intero; il prezzo studenti è il 50%.
   - *Relazione*: associata a uno spettacolo.

10. **Prenotazione**

    - *Attributi*: numero progressivo, data, ora, posto, tipo di biglietto, prezzo.
    - *Regola*: è valida solo se ci sono posti disponibili.

11. **Commento**

    - *Attributi*: pseudonimo, data, ora, testo.
    - *Relazione*: può rispondere ad altri commenti.

12. **Newsletter**

    - *Attributi*: data, ora, oggetto, testo.

Transazioni Tipiche

1. **Inserimenti**

   - Nuova stagione teatrale e relativi interpreti/produttori.
   - Prenotazione di biglietti.
   - Nuovo commento per uno spettacolo.

- Nuova notizia nella newsletter.
- Inserimento/rimozione di un dipendente.

2. **Ricerche e Visualizzazioni**

   - Spettacoli di una stagione teatrale: titolo, descrizione, data, ora, luogo, posti disponibili, prezzi.
   - Attori, registi o autori di una stagione teatrale.
   - Dipendenti di un teatro.
   - Commenti relativi a uno spettacolo.
   - Notizie relative a un teatro.
   - Orari di apertura delle biglietterie.

3. **Statistiche**

   - Andamento stagioni teatrali della rete:
     - Media spettatori paganti.
     - Media affluenza (% tra paganti e capienza).
     - Media incassi.
     - Ordinamento spettacoli per paganti, affluenza, incasso.

Aspetti Critici

1. **Coerenza e Validazione**

   - Le regole sui prezzi dei biglietti e la disponibilità dei posti devono essere rispettate.
   - Le associazioni tra entità (es. teatri, stagioni, spettacoli) devono essere gestite con attenzione per evitare incoerenze.

2. **Efficienza delle Transazioni**

   - L'80% del carico potrebbe derivare da operazioni come prenotazioni e ricerche.
   - Strutture come indici e relazioni pre-calcolate possono migliorare le prestazioni.

3. **Scalabilità**

   - La rete teatrale potrebbe espandersi: il modello deve essere flessibile per aggiungere nuovi teatri, spazi o stagioni.

4. **Interazione con Sistemi Esistenti**

   - La base di dati dovrà integrarsi con eventuali sistemi informatici preesistenti per mantenere continuità operativa.

Ecco una descrizione iniziale del diagramma ER (Entity-Relationship) per la rete teatrale. Ti spiegherò le entità, gli attributi principali e le relazioni.

Entità e Attributi

1. **Teatro**

- Attributi: nome, indirizzo, telefono, fax, email, pagina web.
- Relazione: ha (1:N) Biglietterie, propone (1:N) Stagioni Teatrali.

2. **Biglietteria**

   - Attributi: nome, indirizzo, telefono, email, orari di apertura.
   - Relazione: associata a (N:1) un Teatro.

3. **Dipendente**

   - Attributi: codice fiscale, nome, cognome, residenza, data di nascita, telefono, email, data assunzione, ruolo, stipendio.
   - Relazione: lavora per (N:M) Teatri.

4. **Spazio Teatrale**

   - Attributi: nome, indirizzo, pianta, capienza.
   - Relazione: condiviso da (N:M) Teatri, ospita (1:N) Messe in Scena.

5. **Stagione Teatrale**

   - Attributi: nome, biennio.
   - Relazione: composta da (1:N) Spettacoli, proposta da (N:1) un Teatro.

6. **Spettacolo**

   - Attributi: titolo, anno di produzione, descrizione.
   - Relazione: include (N:M) Interpreti, prodotto da (N:M) Produttori, messo in scena in (1:N) Messe in Scena.

7. **Interprete**

   - Attributi: nome, ruolo, CV.
   - Relazione: partecipa a (N:M) Spettacoli.

8. **Produttore**

   - Attributi: nome, indirizzo, email, telefono (per enti esterni).
   - Relazione: produce (N:M) Spettacoli.

9. **Messa in Scena**

   - Attributi: data, ora, immagini, video, prezzo biglietto (intero, ridotto, studenti), posti disponibili.
   - Relazione: associata a (N:1) uno Spettacolo, ospitata da (N:1) uno Spazio Teatrale, oggetto di (1:N) Prenotazioni.

10. **Prenotazione**

    - Attributi: numero, data, ora, posto, tipo biglietto, prezzo.
    - Relazione: associata a (N:1) una Messa in Scena.

11. **Commento**

- Attributi: pseudonimo, data, ora, testo.
- Relazione: riferito a (N:1) una Messa in Scena, può rispondere a (N:1) un altro Commento.

12. **Newsletter**

- Attributi: data, ora, oggetto, testo.
- Relazione: gestita da (N:1) un Teatro.

Relazioni Principali

1. **Teatro ↔ Spazio Teatrale**
   - Multiplicità: N:M (uno spazio può essere condiviso da più teatri).
2. **Teatro ↔ Dipendente**
   - Multiplicità: N:M (un dipendente può lavorare per più teatri).
3. **Stagione Teatrale ↔ Spettacolo**
   - Multiplicità: N:M (uno spettacolo può far parte di più stagioni).
4. **Spettacolo ↔ Interprete**
   - Multiplicità: N:M (uno spettacolo coinvolge più interpreti e viceversa).
5. **Spettacolo ↔ Produttore**
   - Multiplicità: N:M (uno spettacolo può essere prodotto da più enti).
6. **Spettacolo ↔ Messa in Scena**
   - Multiplicità: 1:N (uno spettacolo può essere messo in scena più volte).
7. **Messa in Scena ↔ Prenotazione**
   - Multiplicità: 1:N (una messa in scena può avere più prenotazioni).
8. **Messa in Scena ↔ Commento**
   - Multiplicità: 1:N (una messa in scena può avere più commenti).

Progettazione Concettuale: Passaggi Principali

1. Formalizzare i requisiti sui dati in un diagramma Entità-Relazione (ER)

- Il primo passo è tradurre i requisiti raccolti in un diagramma ER che rappresenta visivamente le entità, gli attributi e le relazioni.
- **Output**: Un diagramma che include le entità principali, le relazioni tra di esse, le cardinalità, gli attributi e le chiavi primarie.

---

2. Documentare il diagramma

- Per ogni entità, relazione e attributo, occorre fornire una breve descrizione che ne spieghi il significato.
- **Struttura della documentazione**:
    - **Entità**: Nome, descrizione, attributi (specificando chiavi primarie e candidate).
    - **Relazioni**: Nome, descrizione, entità coinvolte, cardinalità, attributi delle relazioni.
    - **Note**: Eventuali dettagli aggiuntivi sulle regole che governano l'uso delle entità o delle relazioni.

---

3. Aggiungere le regole aziendali non codificabili nel diagramma

- Le regole che non possono essere rappresentate nel diagramma ER (come vincoli complessi o logiche condizionali) devono essere documentate separatamente.
- **Esempi**:
    - "Un teatro non può mettere in scena più di due spettacoli di propria produzione all'interno della stessa stagione teatrale."
    - "Il prezzo del biglietto ridotto è sempre l'80% di quello intero, mentre quello per studenti è il 50%."

---

4. Progettare le transazioni

- Ogni transazione identificata nei requisiti deve essere progettata in dettaglio. Questo include:
    - Input richiesti (dati da inserire).
    - Output generati (risultati attesi).
    - Logica di elaborazione (passaggi da eseguire per ottenere l'output).
- **Esempi di transazioni progettate**:
    - **Inserire una nuova stagione teatrale**: Specificare i dati da inserire (nome stagione, biennio, elenco spettacoli) e verificare che non esistano conflitti (es. duplicazioni).
    - **Trovare gli spettacoli di una stagione**: Richiedere il nome della stagione e restituire i dettagli degli spettacoli in ordine cronologico.

---

5. Verificare la qualità degli schemi concettuali prodotti

- **Criteri di verifica**:
    - **Completezza**: Tutti i requisiti sono rappresentati nel diagramma o nella documentazione?
    - **Chiarezza**: Gli schemi e la documentazione sono facilmente comprensibili?
    - **Consistenza**: Non ci sono contraddizioni o ridondanze tra entità e relazioni.
    - **Flessibilità**: Gli schemi sono abbastanza generici da supportare futuri cambiamenti?
    - **Efficienza**: Le regole progettate ottimizzano la gestione delle transazioni tipiche.

La progettazione dei dati basata sul modello **Entità-Relazione (ER)** è un passaggio fondamentale per creare un sistema coerente e ben strutturato. Il processo include la creazione del **diagramma ER**, la **documentazione** associata e la definizione di eventuali **regole aziendali**.

Componenti del modello ER

1. Diagramma ER

Il diagramma ER rappresenta visivamente le entità, gli attributi e le relazioni tra le entità.

- **Entità**: Oggetti o concetti del mondo reale che si vogliono rappresentare (es. Teatro, Spettacolo, Dipendente).
- **Relazioni**: Associazioni tra le entità (es. Un Teatro organizza Spettacoli).
- **Attributi**: Proprietà che descrivono le entità e le relazioni (es. Nome e Indirizzo per Teatro).
- **Chiavi primarie**: Attributi che identificano in modo univoco ogni istanza di un'entità.

---

2. Documentazione

Ogni componente del diagramma deve essere accompagnata da una descrizione in linguaggio naturale:

- **Entità**:
    - **Nome**: Identifica l'entità.
    - **Descrizione**: Spiega il significato dell'entità.
    - **Attributi**: Specifica i dati associati all'entità, evidenziando la chiave primaria e altre caratteristiche (es. attributi derivati, multivalore).
- **Relazioni**:
    - **Nome**: Indica il tipo di relazione.
    - **Descrizione**: Spiega la connessione tra le entità coinvolte.
    - **Partecipazione e cardinalità**: Specifica il numero minimo e massimo di partecipanti di ciascun lato della relazione.

---

3. Regole aziendali

Alcuni vincoli sui dati non possono essere rappresentati graficamente nel diagramma ER. Questi vincoli vengono definiti come regole aziendali e documentati separatamente.
Esempi:

- Un teatro non può avere più di due spettacoli di propria produzione in una stagione teatrale.
- Il prezzo del biglietto ridotto è sempre l’80% del prezzo intero.

Esempio per la rete teatrale

Entità

1. **Teatro**
    - **Descrizione**: Un teatro che fa parte della rete.
    - **Attributi**:
        - Nome (PK)

        - Indirizzo
        - Telefono
        - Email
        - Sito Web

2. **Spettacolo**
    - **Descrizione**: Un evento teatrale proposto da un teatro.
    - **Attributi**:
        - Titolo (PK)
        - Anno di produzione
        - Descrizione
        - Genere

Relazioni

1. **Organizza** (Teatro → Spettacolo)
    - **Descrizione**: Un teatro organizza uno o più spettacoli.
    - **Cardinalità**:
        - Un teatro organizza da 0 a molti spettacoli.
        - Uno spettacolo può essere organizzato da uno o più teatri.

Il diagramma **Entità-Relazione (ER)** è un potente strumento di progettazione concettuale utilizzato per rappresentare la struttura di una base di dati in modo chiaro e comprensibile. Esso include entità, relazioni, attributi e regole aziendali, ognuno con una rappresentazione grafica specifica, che aiuta a visualizzare come le informazioni sono collegate tra loro.

Ecco una panoramica delle principali componenti del diagramma ER:

1. Entità

Le entità rappresentano concetti di rilevo che hanno un'esistenza autonoma nel sistema. Ogni entità ha un **nome univoco** e può avere diversi attributi. Le entità sono rappresentate nel diagramma con **rettangoli**. Ad esempio:

- **Teatro** (Entità)
- **Spettacolo** (Entità)
- **Dipendente** (Entità)

2. Relazioni

Le relazioni collegano due o più entità tra di loro. Ogni relazione ha un **nome univoco** ed è rappresentata nel diagramma con un **rombo**, con linee che connettono la relazione alle entità coinvolte. Le relazioni possono essere:

- **Uno a molti**: Una entità è associata a una o più istanze di un'altra entità, ma ogni istanza dell'altra entità è associata a una sola istanza della prima.
- **Molti a molti**: Entrambe le entità sono associate a molteplici istanze dell'altra.
- **Uno a uno**: Ogni istanza di una entità è associata esattamente a una istanza dell'altra entità.

Inoltre, possono esserci **relazioni ricorsive** (che legano la stessa entità) o **relazioni ternarie** (che coinvolgono più di due entità).

3. Attributi

Gli attributi sono le proprietà che descrivono un'entità o una relazione. Ogni attributo ha un nome univoco ed è rappresentato con un **ellisse**. Gli attributi possono essere:

- **Semplici**: Ogni entità ha un valore unico (es. nome, indirizzo).
- **Composti**: Sono formati da più attributi (es. indirizzo composto da via, numero, città).
- **Multivalore**: Ogni entità può avere più valori per un attributo (es. numeri di telefono).
- **Calcolati**: Il valore dell'attributo è derivato da altri attributi (es. incasso totale calcolato da numero di spettatori e prezzo).

4. Chiavi

Le **chiavi primarie** sono gli attributi che identificano univocamente un'istanza di un'entità. Vengono rappresentate con il nome dell'attributo **sottolineato**. Ogni entità deve avere una chiave primaria, che può essere composta da più attributi. Se un'entità non ha una chiave autonoma, può essere una **entità debole**, che dipende da un'altra entità per la sua identificazione.

5. Specializzazioni

Le **specializzazioni** consentono di creare sotto-entità che ereditano attributi e relazioni dall'entità principale (genitore). Questo è simile al concetto di sotto-classe nei linguaggi di programmazione orientati agli oggetti. Una specializzazione può essere:

- **Totale**: Ogni istanza dell'entità genitore appartiene a una sotto-entità.
- **Parziale**: Solo alcune istanze dell'entità genitore sono specializzate.
- **Disgiunta**: Ogni istanza dell'entità genitore appartiene a una sola sotto-entità.
- **Sovrapposta**: Un'istanza dell'entità genitore può appartenere a più di una sotto-entità.

6. Relazioni Identificanti

Se un'entità debole è identificata tramite una relazione con un'entità proprietaria, la relazione è detta **identificante**. In questo caso, la partecipazione dell'entità debole alla relazione è obbligatoria (1,1).

Esempio di Diagramma ER per una Rete Teatrale

1. **Entità**:
    - **Teatro**: Nome, Indirizzo, Telefono
    - **Spettacolo**: Titolo, Anno di Produzione
    - **Dipendente**: Nome, Ruolo, Data di Assunzione
    - **Attore**: Nome, Cognome, Data di Nascita
2. **Relazioni**:
    - **Organizza**: Relazione tra **Teatro** e **Spettacolo** (Un teatro può organizzare molti spettacoli).
    - **Recita**: Relazione tra **Attore** e **Spettacolo** (Un attore può recitare in più spettacoli).

- **Lavora presso**: Relazione tra **Dipendente** e **Teatro** (Un dipendente lavora in un teatro).

3. **Attributi**:

   - **Teatro**: Nome (chiave primaria), Indirizzo, Telefono, Email
   - **Spettacolo**: Titolo (chiave primaria), Anno di Produzione
   - **Attore**: Nome (chiave primaria), Cognome, Data di Nascita

4. **Cardinalità**:

   - Un **Teatro** può organizzare molti **Spettacoli** (relazione molti a molti).
   - Un **Attore** può recitare in molti **Spettacoli** (relazione molti a molti).
   - Un **Dipendente** lavora presso un **Teatro** (relazione uno a molti).

Questo è un esempio di come un diagramma ER potrebbe essere progettato per una rete teatrale. Ogni componente del sistema (Teatro, Spettacolo, Dipendente, Attore) è ben definito, con attributi chiave e relazioni chiare che collegano tra loro le entità.

La progettazione delle transazioni è un passo cruciale per garantire che il sistema di database risponda efficacemente ai requisiti funzionali. Ecco come questa fase si articola in dettaglio:

1. **Classificazione delle Transazioni**

Le transazioni vengono suddivise in tre categorie principali:

- **Transazioni di aggiornamento**:

  - Queste operazioni modificano il contenuto del database. Possono essere:
    - **Inserimento** (ad esempio, aggiungere una nuova istanza di un'entità come un nuovo teatro o un attore).
    - **Modifica** (ad esempio, aggiornare l'indirizzo di un dipendente).
    - **Cancellazione** (ad esempio, rimuovere un dipendente dal sistema).

- **Transazioni di interrogazione**:

  - Operazioni che consentono di ottenere informazioni senza modificarle. Sono orientate al recupero dei dati. Ad esempio, ottenere una lista di spettacoli programmati in un determinato teatro o visualizzare i dettagli di un attore.

- **Transazioni miste**:

  - Combinano sia l'aggiornamento che l'interrogazione dei dati. Ad esempio, un'operazione che aggiorna lo spettacolo in corso e restituisce la lista degli attori associati a quello spettacolo.

2. **Specifiche della Transazione**

Ogni transazione deve essere descritta in modo completo, dettagliando i seguenti aspetti:

- **Dati di Input**:

    - Gli oggetti di dati che la transazione richiede. Questi dati provengono da interazioni con l'utente o da altri sistemi. Per esempio, in una transazione di inserimento, l'input potrebbe includere i dati relativi a un nuovo attore (nome, cognome, età).

- **Dati di Output**:
    - I risultati prodotti dalla transazione. Ad esempio, in una transazione di interrogazione, l'output potrebbe essere un elenco di tutti gli spettacoli programmati in un teatro.

- **Comportamento Funzionale**:
    - Una descrizione ad alto livello di ciò che la transazione fa. Questo deve essere indipendente dalla tecnologia o dal linguaggio di programmazione utilizzato. Ad esempio, una transazione di aggiornamento potrebbe descrivere come un nuovo attore viene associato a uno spettacolo, inclusi i controlli di integrità e validità dei dati.

3. **Verifica di Fattibilità**

Una parte essenziale della progettazione delle transazioni è la verifica della loro fattibilità rispetto allo schema concettuale dei dati. È importante che ogni transazione possa essere eseguita correttamente sul modello ER, al fine di evitare problemi di progettazione.

- **Verifica sul modello concettuale**:
    - Ogni transazione deve essere tracciata attraverso il diagramma ER per assicurarsi che tutte le operazioni siano compatibili con la struttura dati definita. Per esempio, se una transazione implica l’aggiornamento di un’entità, occorre verificare che la chiave primaria e le relazioni siano gestite correttamente.

- **Controllo delle regole aziendali**:
    - Le regole aziendali, che stabiliscono le modalità di gestione delle informazioni (ad esempio, un attore non può essere associato a più di un teatro contemporaneamente), devono essere rispettate durante l'esecuzione delle transazioni. Se la transazione violerebbe una regola aziendale, è necessario adattare lo schema o modificare la transazione stessa.

4. **Implementazione delle Transazioni**

Dopo la progettazione e la verifica, le transazioni vengono implementate. Tuttavia, questa fase dovrebbe rimanere indipendente dal sistema di gestione del database (DBMS) specifico, evitando di legarsi a dettagli di implementazione. Questo approccio consente di mantenere la progettazione concettuale chiara e astratta, permettendo eventualmente l'adattamento a diversi sistemi o tecnologie.

Conclusione

La progettazione delle transazioni è fondamentale per garantire che un database possa eseguire operazioni in modo efficiente e corretto. La definizione chiara delle transazioni aiuta a identificare eventuali problemi nel modello dei dati e consente di assicurare che tutte le operazioni siano possibili senza violare regole aziendali o limitazioni del sistema.

La **verifica della qualità** di uno schema concettuale è un passaggio fondamentale per assicurarsi che il modello creato rispetti gli standard richiesti e possa essere tradotto efficacemente in uno schema logico, che sarà poi implementato in un sistema di gestione di basi di dati. Le proprietà principali da verificare in questa fase sono:

1. **Correttezza**

- **Definizione**: Uno schema è corretto quando utilizza i costrutti del modello concettuale in modo appropriato e conforme alle regole del modello stesso.
- **Verifica**:
    - I concetti (entità, relazioni, attributi) devono essere definiti correttamente, rispettando le regole sintattiche e semantiche del modello.
    - Le relazioni tra le entità devono essere definite in modo che rispecchino correttamente i legami tra i dati.
    - Le cardinalità (ad esempio, 1:N, N:M) e i vincoli (ad esempio, chiavi primarie, chiavi esterne) devono essere applicati correttamente.
- **Esempio**: Se in un sistema di gestione di teatri si modella una relazione tra attori e spettacoli, la cardinalità deve essere definita correttamente, come “un attore può essere associato a più spettacoli, ma uno spettacolo deve avere almeno un attore”.

2. **Completezza**

- **Definizione**: Uno schema è completo quando rappresenta **tutti** i requisiti identificati nella fase di raccolta e analisi dei requisiti. Inoltre, tutte le transazioni e le regole aziendali devono poter essere eseguite sulla base dello schema concettuale.
- **Verifica**:
    - **Requisiti dei dati**: Tutte le informazioni necessarie devono essere rappresentate nel modello. Non devono mancare entità, attributi o relazioni che siano stati richiesti dal cliente o dai soggetti interessati.
    - **Transazioni**: Ogni tipo di transazione identificata durante la fase di progettazione delle transazioni deve poter essere eseguita nel contesto dello schema concettuale senza errori o impossibilità operative.
    - **Regole aziendali**: Lo schema deve garantire che le regole aziendali siano rispettate e possano essere applicate durante le transazioni, come ad esempio vincoli di integrità, limitazioni sul tipo di dato o restrizioni sulle relazioni.
- **Esempio**: Se un requisito aziendale prevede che ogni spettacolo abbia una data di inizio e fine, lo schema deve includere un attributo "data_inizio" e "data_fine" per ogni entità spettacolo.

3. **Minimalità**

- **Definizione**: Uno schema è minimale quando non è possibile eliminare alcun concetto dallo schema senza compromettere la sua completezza. Cioè, lo schema non deve contenere concetti ridondanti o superflui.
- **Verifica**:
    - **Eliminazione di concetti**: Ogni entità, relazione o attributo deve avere una ragione d'essere nel modello. Non devono esserci elementi inutilizzati o ridondanti che non contribuiscono a soddisfare i requisiti del sistema.

    - **Ridondanza**: Se uno schema contiene più di una rappresentazione di un concetto simile, occorre rimuovere o combinare questi concetti per evitare duplicazioni.
- **Esempio**: Se un modello contiene due entità distinte per "Attore" e "Persona" che condividono gli stessi attributi, una di queste entità potrebbe essere ridondante. Si dovrebbero combinare, mantenendo solo le informazioni necessarie.

Conclusione

La verifica di qualità di uno schema concettuale è un passo cruciale prima di passare alla fase successiva della progettazione logica. Un buon schema concettuale:

- **Corretto**, garantendo l'uso adeguato dei costrutti del modello;
- **Completo**, rappresentando tutte le necessità aziendali e permettendo l'esecuzione di tutte le transazioni;
- **Minimale**, evitando concetti inutili o ridondanti.

Solo dopo aver raggiunto uno schema concettuale di alta qualità si può procedere con la progettazione logica, che tradurrà il modello concettuale in uno schema fisico e implementabile su un DBMS.