Basi di Dati

Leonardo Valente

April 8, 2022

Contents

1 Modello ER

Qualsiasi progetto, prima di essere mandato in produzione, segue un ciclo di vita, di solito composto da:

- Studio di fattibilità: definizione dei costi e delle priorità
- Raccolta di analisi e dei requisiti: studio delle proprietà del sistema
- Progettazione: di dati e funzioni
- Implementazione: ovvero la realizzazzione del progetto
- Validazione e collaudio: la fase di sperimentazione
- Funzionamento: fase di produzione
- Manutenzione: dove arriva il vero guadagno

Questo ciclo di vita segue un **modello a spirale**, quindi un ciclo dove in ogni fase è possibile andare avanti o indietro in base alle esigenze.

1.1 Progettazione

La progettazione delle applicazioni schematizza le operazioni sui dati e progetta il software. E' opportuno quindi seguire una **metodologia di progetto**. Essa ci permette di **suddividere** la progettazione in fasi indipenti, fornendo delle **strategie** da seguire e dei **criteri** di scelta.
Nella progettazione di Database ci sono 3 fasi di progettazione:

- Progettazione concettuale (modello ER)
- Progettazione logica
- Progettazione fisica

Ognuna di queste fasi si basa su un modello che permette di generare una rappresentazione formale (di solito uno schema) del nostro universo.

1.1.1 Progettazione concettuale

Traduce i requisiti del sistema in un modello ER, espresso in modo indipendente dalle scelte implementative.
La descrizione si deve concentrare sui **dati** e sulle loro **relazioni**, non sulle scelte implementative.

1.1.2 Progettazione logica

Consiste nella traduzione dello schema concettuale nel modello dei dati del DBMS (Modello relazionale).

1.1.3 Progettazione fisica

Completa lo schema logico ottenuto con le specifiche proprio dell'HW/SW scelto. Viene interamente effettuato dal DBMS.

1.2 Introduzione al Modello ER

Il modello Entità - Relazione è un **linguaggio grafico semi-formale** per la rappresentazione di schemi concettuali. Esso è ormai diventato uno *standard* nelle metodologie di progetto.

Iniziamo facendo una distinzione fra **Entità** e **Relazioni**.

- **Entità**: classe di oggetti dell'applicazione di interesse con proprietà comuni e con esistenza "autonoma", della quale si vogliono registrare fatti specifici.
  Le entità hanno degli attributi che la descrivono.
  Una **occorrenza** (o istanza) di una entità è il singolo oggetto creato sulla base dell'entità da cui deriva
  Le entità vengono rappresentate nel modello ER tramite dei *rettangoli*.

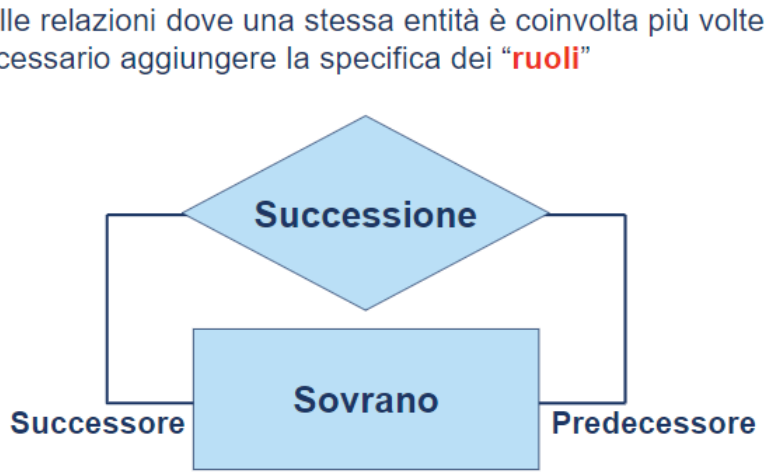
- **Attributi**: un attributo è definito su un dominio di valori. Esso è una funzione che associa ad ogni occorrenza **un** particolare valore (non di più!) Gli attributi possono essere composti e possono essere **qualsiasi cosa**.
  A meno che non venga definito il contrario, gli attributi sono **obbligatori**.

- **Relazione**: fatto che descrive un'azione o una situazione e che *stabilisce legami logici tra istanze di entità*.
  I legami possono essere fra più di due entità e il numero delle entià coinvolte ne determina il **grado**.
  Ogni relazione ha un nome che la identifica.
  Inoltre, le relazioni possono avere degli attributi, chiamati appunto **attributi delle relazioni** che modellano il legame tra le entità.

  Le relazioni vengono rappresentate nel modello ER tramite dei *rombi*.

Associazioni ad anello

Un'associazione può coinvolgere "due o più volte" la stessa entità (associazione ricorisiva o ad anello).



2 Cardinalità delle Relazioni

La cardinalità delle relazioni non è altro che una coppia di valori che si associa a ogni entità che partecipa ad una relazione. Esprimono un **limite minimo** (cardinalità minima) e un **limite massimo** (cardinalità massima) di istanze della **relazione R** a cui può partecipare ogni istanza dell'**entità E**.
E' molto importante non sbagliare la *cardinalità massima*. E' la più importante fra le due. Se sbagliamo la cardinalità, sbagliamo il Database!

Nota: anche gli attributi possono avere cardinalità!

- **Molti a Molti** (3 tabelle). Quando zero o più istanze della prima entità si possono associare a zero o più istanze della seconda entità.
- **Uno a Molti** (2 tabelle). Quando ogni istanza della prima entità si può associare a una o più istanze della seconda entià.
- **Uno a Uno** (2 tabelle). Quando ogni istanza della prima entità si può associare a una e una sola istanza della seconda entià.

2.1 Identificatori di una entità

E' uno strumento per l'identificazione univoca delle occorrenze di una entità.
Ci sono due tipi di identificatori:
(!!!! CAMBIARE LE DEFINZIONI NON SONO CORRETTE !!!)

- **Identificatore interno** (Primary Key), che serve per distinguere in modo univoco una istanza di una entità. *Deve essere unico.*
  L'identificatore interno può essere combinazione di più attributi!
- **Identificatore esterno** (Foreign Key), che serve per fare riferimento ad una istanza di un'altra entità avente come identificatore primario l'identificatore esterno dell'oggetto che stiamo prendendo in considerazione.

Ogni entità deve possedere almeno un identificatore (primario), ma può averne in generale più di uno (esterno).

3 Relazione IS-A

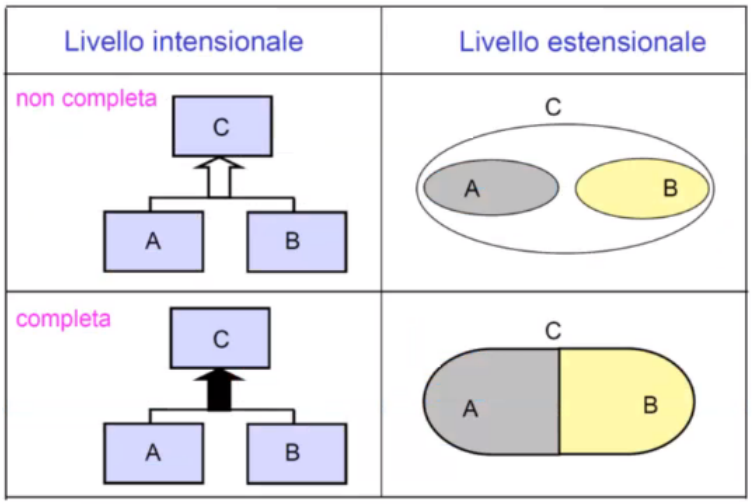
Può accadere che tra due classi rappresentate da due entità nello schema concettuale sussista la **relazione IS-A**, cioè che **ogni istanza di una sia anche istanza dell'altra**.
La relazione IS-A si può definire tra *due entità*: **entità padre** e **entità figlio**.

3.0.1 Generalizzazione

Può capitare però, che l'entità padre può generalizzare diverse sottoentità rispetto ad un unico criterio. In questo caso si parla di **generalizzazione**.
Una generalizzazione può essere di *due tipi*:

- **Completa**: l'unione delle istanze delle sottoentità è uguale all'insieme delle istanze dell'entità padre.
- **Non completa**.

approccio prevede la distinzione logica tra entità comuni e non, come uomo/donna e sportivo o impiegato.



Un ulteriore caratteristica della generazione è l'esclusività, una generalizzazione esclusiva prevede che un elemento possa appartenere solo a una delle categorie (uomo o donna). In caso vi siano più appartenenze, si definisce sovrapposto, ad esempio uomo donna e dottore (un'istanza può essere sia dottore che uomo o donna).
Come possiamo vedere, nella relazione *completa*, l'unione dei due sottoinsiemi A e B è proprio l'insieme "padre" C. Nella relazione *NON completa*, invece, ci saranno alcuni elementi di C che non fanno parte ne di A e ne di B.
Una entità può avere **al massimo una e una sola** entità padre.

3.0.2 Principio di ereditarietà

Ogni proprietà del padre è anche una proprietà del figlio, che però non viene esplicitamente riportata nello schema concettuale.
L'entità figlio può avere degli attributi in più rispetto all'entità padre.

3.1 Strategie di Progetto

Come procediamo con tante specifiche anche dettagliate?

- top-down
- bottom-up
- inside-out

Top-down

Si parte da uno schema iniziale molto astratto ma completo, che viene successivamente raffinato fino ad arrivare allo schema finale.
Si parte dall'alto, e mano a mano si scende

Bottom-up

Si suddividono le specifiche in modo da sviluppare semplici schemi parziali ma dettagliati, che poi vengono integrati tra di loro

Inside-out

Parto dal primo termine e mano a mano sviluppo.

Esiste anche una **strategia ibrida**, che è un insieme di tutte e 3.

Lo schema scheletro facilita le fasi di integrazione

- Si individuano i concetti più importanti (i più citati o quelli indicati come cruciali) (BOTTOM UP)
- Si organizzano tali concetti in un semplice schema concettuale (INSIDE OUT)
- Ci si concentra sugli aspetti essenziali (molti attributi, le cardinalità delle relazioni, le gerarchie articolate sono rimandate) (TOP DOWN)

E' la strategia più flessibile perché permette si suddividere il problema in sottoproblemi e di procedere per raffinamenti progressivi

4 Modello Relazionale

Un modello dei dati è un insieme di concetti per organizzare i dati e descriverne la struttura. Questo modello deve essere comprensibile a un elaboratore. La componente fondamendale di ogni modello sono i **meccanismi di strutturazione**.

Il **Modello Relazionale** è il modello di dati più diffuso.

Permette di definire per mezzo del costruttore relazione che permette di organizzare i dati in insiemi di record a struttura fissa. Una relazione è spesso rappresentata da una tabella. *La relazione è un modo di relazionare i dati. Non è una relazione matematica ne è una relazione del modello ER.*

nome	cognome	Data di nascita	professione	tel
mario	rossi	21/10/80	impiegato	02 345678
sara	bianchi	17/03/77	avvocato	031 45678
marco	verdi	11/11/67	medico	06 789052

Il Livello Logico è **indipendente** da quello fisico: una tabella è usata nello stesso modo sia nel livello logico che nel livello fisico.

In una relazione non ci sono più tuple con lo stesso valore. Altrimenti non sarebbe una relazione.

Definizione di *Schema di Relazione*

Un nome R con un insieme di attributi $X = \{A_1, A_2, ..., A_n\}$
es: STUDENTI(Matricola, Cognome, Nome, DataNascita)

Definizione di *Schema di Base di Dati*

Insieme di schemi di relazione con nomi diversi:
R = STUDENTI(...), ESAMI(...), CORSI(...)
R = $\{R_1(X_1), ..., R_n(X_n)\}$

Definizione di *Tupla*

Una tupla su un insieme di attributi X è una funzione t che associa a ciascun attributo A in X un valore del dominio di A.
t[A] oppure t.A denota il valore della tupla t sull'attributo A.

Chiave

Insieme di attributi che identificano univocamente le tuple di una relazione. Formalmente:

- Un insieme K di attributi è **superchiave** per r se r non contiene due tuple distinte t_1 e t_2 tali che $t_1[K] = t_2[K]$.
- K è **chiave** per r se è una superchiave minimale per r
  (minimale significa che se si dovesse eliminare un attributo della superchiave, essa non risulterebbe più una superchiave)

L'esistenza delle chiavi garantisce l'accessibilità a ciascun dato della base di dati.

All'interno di una relazione, però, ci possono essere dei valori *nulli*. Tre casi differenti:

- Valore sconosciuto
- Valore inesistente
- Valore senza informazione

I DBMS non fanno distinzione tra questi casi.

4.1 Vincoli di integritrà

Alcuni esempi di vincoli di integrità:

- Valori nulli
- Valori fuori del dominio
- Tuple incosistenti
- Tuple con valori uguale per chiavi
- Valori inesistenti in attributi usati per corrispondenze tra relazioni

I vincoli di integrità vengono usati per descrivere in modo più accurato la realtà dei fatti. Danno un contributo alla "qualità dei dati" e sono utili nella progettazione. Sono anche usati dai DBMS nella esecuzione delle interrogazioni.

Alcune volte, però, non tutte le proprietà di interesse sono rappresentabili per mezzo di vincoli di integrità esprimibili direttamente. Verrano gestiti al di fuori della base di dati.

Tipi di vincoli

- Vincoli su **valori** (un voto di esame non può andare oltre a 30)
- Vincoli di **tupla** (una lode può essere data solo a un voto di valore 30)
- Vincoli di **chiave** (non possono esistere due chiavi)

  - Vincolo di dominio. Es.:
    (Voto $\geq$ 18) AND (Voto $\leq$ 30)

  - Vincolo di tupla. Es.:

    (Voto = 30) OR (NOT (Lode = "e lode"))

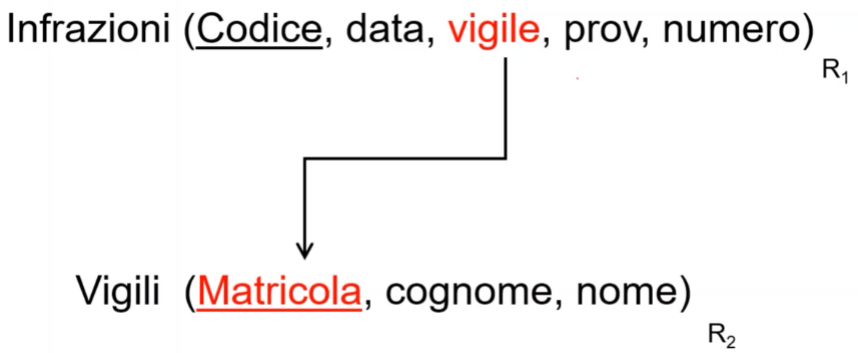
Esistono anche dei *Vincoli* ***interrelazioni*** che coinvolgono più relazioni:

- Vincoli di integrità **referenziale**

4.2 Vincoli di Integrità Referenziale

Un vincolo di integrità referenziale è un vincolo di integrità **interrelazionale**, ovvero si verifica quando andiamo a trattare due o più relazioni.

Una **Foreign Key** (vincolo di integrità referenziale) fra gli attributi X (uno o più attributi) di una relaizone R_1 e un'altra relazione R_2 impone ai valori su X in R_1 di comparire **come valori della chiave primaria di** R_2.



Nel compitino ***Le foreign key vanno cerchiate!!***

5 Progettazione Logica

6 SQL

Structured Query Language è un linguaggio dichiarativo per la definizione e la manipolazione dei dati in database relazionali, adottato da molti DBMS.

SQL è **relazionalmente completo**: ogni espressione dell'algebra relazionale può essere tradotta in SQL. Esso adotta la logica a 3 valori (T, F, U).
Il modello dei dati di SQL è basato su **tabelle** anzichè **relazioni**.

SQL contiene:

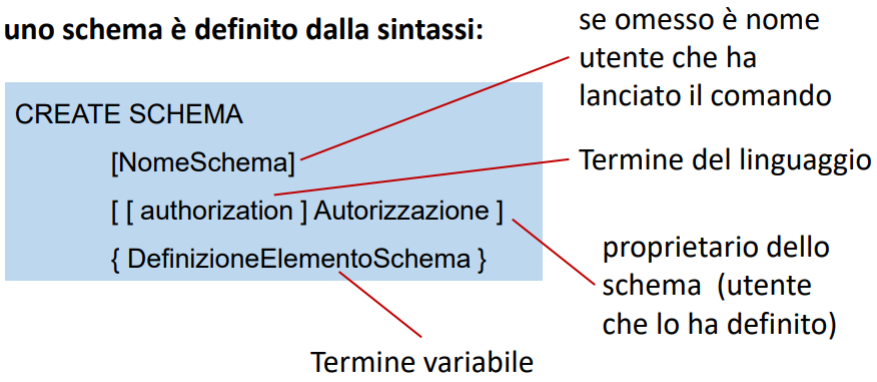
- DDL (Data Definition Language) - operazioni di definizione e modifica di schemi
- DML (Data Manipulation Language) - operazioni di inserimento, modifica e cancellazione di dati

Istruzioni Principali di DML

- SELECT - Operazioni di interrogazione. Formula query come quelle dell'AR o richieste più elaborate.
- INSERT, DELETE, UPDATE - Operazioni di aggiornamento. Inserisce, elimina, aggiorna tuple.

Istruzioni Principali di DML

- CREATE SCHEMA - Crea un nuovo schema



- CREATE TABLE - Crea una nuova tabella;
  Una tabella è costituita da una collezione ordinata di attributi e da un insieme (eventualmente vuoto) di vincoli.

```
CREATE TABLE NomeTabella (
        NomeAttributo Dominio [ ValoreDiDefault ] [Vincoli]
        { NomeAttributo Dominio [ ValoreDiDefault ] [Vincoli] }
        [AltriVincoli] )
```

Domini Elementari

SQL ha 6 domini predefiniti

- Carattere (VARCHAR, CHAR)
- Numerico Esatto (INTEGER)
- Numerico Approssimato
- Data/Ora (TIMESTAMP, DATE, TIME)
- Intervallo Temporale
- Boolenan (ex Bit)

C'è anche la possibilità di definire nuovi domini usando **CREATE DOMAIN**.

```
CREATE DOMAIN NomeDominio AS DominioElementare
    [ ValoreDefault ]
    [ Constraints ]
```

Questo comando definisce un nuovo dominio elementare utilizzabile in definizioni di relazioni.

```
CREATE DOMAIN Voto AS SMALLINT
    DEFAULT 0
    CHECK (value >= 18 AND value <= 30)
```

Vincoli

Un vincolo è una regola che specifica delle condizioni sui valori di un elemento dello schema del database.

Un vincolo può essere associato ad una tabella, attributo o dominio.

- Vincoli Intrarelazionali - proprietà sempre valida all'interno di una relaizone
  - NOT NULL - il valore deve essere non nullo
  - UNIQUE - i valori devono essere non ripetuti
  - PRIMARY KEY - chiave primaria della tabella
  - CHECK - definisce condizioni complesse
- Vincoli Interrelazionali - proprietà sempre valida tra relazioni diverse
-
  - FOREIGN KEY, REFERENCES - permettono di definire le chiavi esterne.
    Impone ai valori degli attributi X nella tabella Figlio di comparire *come valori della chiave primaria della tabella Padre*
  - CHECK

Se viene eseguita una operazione di aggiornamento che porta alla violazione di un vincolo di integrità referenziale, (per ogni vincolo visto) lo stato della base di dati diventa non valido.Sono perciò definite un insieme di politiche per evitare che questo accada. Tali politiche vanno dichiarate nella dichiarazione DDL dello schema.

- CASCADE - l'operazione di modifica/cancellazione viene propagata alla tabella interna
- SET NULL - NULL nella tabella figlio in entrambi i casi
- SE DEFAULT - default nella tabella figlio in entrambi i casi
- NO ACTION - rifiutata in entrambi i casi

6.1 Interrogazioni SQL

Le interrogazioni di SQL avvengono tramite l'istruzione **SELECT** (che non significa "seleziona").