

Gestione dei Big Data

Matteo Aprile

Professore: Marco Zappatore, Antonella Longo

INDICE

| | | |
|------------|--|----------|
| I | Libri di testo consigliati | 1 |
| II | Databases - 28.09.22 | 1 |
| II-A | Definizioni di base | 1 |
| II-B | Tipologie di DB | 1 |
| II-C | Ciclo di vita del DB | 1 |
| II-D | Livelli di un DB | 2 |
| II-E | Data Base Managment System | 2 |
| II-F | Mini-world | 2 |
| III | Database System Concepts and Architecture - 28.09.22 | 2 |
| III-A | Definizioni sui modelli | 2 |
| III-B | Definizioni fondamentali | 2 |
| III-C | Schema | 3 |
| III-D | Tipologie di DBMS | 3 |
| IV | Data Modeling Using the Entity-Relationship (ER) Model - 28.09.22 | 3 |
| IV-A | Entity-Relationship (ER) | 3 |
| V | The Enhanced Entity-Relationship (EER) Model - 04.10.22 | 4 |
| V-A | Superclassi e sottoclassi | 4 |
| V-B | Graficazione superclassi e sottoclassi | 4 |
| | Riferimenti bibliografici | 5 |

I. LIBRI DI TESTO CONSIGLIATI

- Fundamental of Database Systems, 7th ed, Elmasri, Navathe
- Data Warehouse Design, Rizzi, Golfarelli big dataL concepts technology and architecture 1st ed balusamy abirami gadomi

II. DATABASES - 28.09.22

A. Definizioni di base

Le definizioni di base da sapere sono:

- **dato**: insieme di fatti conosciuti, registrati e con un **significato**. È detto dato grezzo visto che si suppone che andrò ad elaborarlo, questo dato **sara' poi archiviato**, sarà un **fatto conosciuto** cioè avremo:
 - **eventi con un significato** per un dato tipologia di utenti
 - sorgente che **produce i dati** con una certa velocità
- **DataBase**: raccolta di dati altamente organizzati, intercorrelati e strutturati. È una struttura con dei collegamenti strutturati tra i dati
- **DBMS Data Base Managment System**: insieme di **programmi per accedere ai dati e farci delle operazioni** di 4 tipi: creazione, recupero, aggiornamento e cancellazione, ciclo **CRUD**. Ne favorisce anche il mantenimento.
- **mini-world**: parte del mondo reale alla quale si riferiscono i dati presi andando a **limitare la modellazione** in un numero n di concetti
- **DataBase System**: insieme di DBMS con i dati
- **astrazione**: **separare i dati dai collegamenti tra le entità per disporle in un modello** senza che esso si occupi di come salvare i dati
- **modello concettuale**: formato da entità e relazioni
- **modello fisico**: definizione dei tipi dato e dove sono **conservati**
- **controllo della concorrenza**: garantire che tutte le transazioni sono **correttamente eseguite**
- **recovery**: se la transazione è stata eseguita è stata conservata nel database

B. Tipologie di DB

Esistono molti tipi di DB:

- numerici o testuali
- multimediali
- Geographic Information Systems (GIS)
- Data Warehouses

C. Ciclo di vita del DB

È opportuno vedere un **concetto di base dei dati**, cioè il loro ciclo di vita. Il più semplice è:

- 1) **acquisizione** (scattered data)
- 2) **aggregazione** (integrated data)
- 3) **analisi** (knowledge)

4) finisce in un **applicazione** che genera dei "log data" che saranno poi acquisiti come scattered data

Da un **punto di vista computazionale** queste fasi si devono prendere in un altro modo:

- 1) storage dei data
- 2) formattazione e pulizia
- 3) capire cosa dicono i dati
- ?) se non mi bastano i dati che ho posso integrare dei dati

D. Livelli di un DB

Quando si ha un DB abbiamo 3 livelli da considerare

- 1) **fisico**: dove sono **salvati i dati**
- 2) **logico**: indica come i dati sono **collegati tra loro**
- 3) **view**: **rappresentazione** che sarà diversa per ogni tipo di utente

E. Data Base Management System

Un DBMS offre l'opportunità di:

- **salvataggio** dei dati
- **definizione** modelli dati
- **manipolazione** dei dati
- **processare** e condividere i dati

Per quanto riguarda l'**interazione con i DB** avremo 2 strumenti:

- **query**: accede a parti differenti di dati e formula una richiesta
- **transazioni**: legge dei dati ed aggiorna alcuni valori, salvandoli nel DB

F. Mini-world

Avremo bisogno di **identificare delle entità**, cioè i **concetti di base** che rappresentano una parte delle cose che inseriremo nel DB relazionale. Poi andremo a **connettere tra loro le entità**, dette relazioni (**relationships**) (ER), **ne derivano delle tabelle dette relation**.

Il tutto **da derivare dai requisiti** e non dall'esperienza personale.

Le tabelle create dalle entità conterranno i dati che ho a disposizione. Saranno divisi in:

- righe (record)
- colonne (attributi)
- celle (dati grezzi)

Si verrà quindi a creare un **catalogo** con vincoli, tipo di dati e la relazione di appartenenza degli attributi.

III. DATABASE SYSTEM CONCEPTS AND ARCHITECTURE - 28.09.22

A. Definizioni sui modelli

Le definizioni di base da sapere sono:

- **Data Model**: insieme di **concetti che descrivono struttura, operazioni e vincoli** applicati al DB
- **Data Model Structure and Constraints**: abbiamo dei **costrutti che definiscono come collegare gli elementi** definiti da: entità, record e tabella
- **Data Model Operation**: di base (**CRUD**) o definite dall'utente
- **modello dal concettuale**: di **alto livello** e semantico
- **modello fisico**: di basso livello, definisce **come i dati sono salvati**
- **modello implementativo**: usati nel DBMS
- **modello autodescrivente**: basati su XML

B. Definizioni fondamentali

- **DataBase schema**: descrizione del database in termini di struttura, tipo dati e vincoli
- **schema diagram**: **visione rappresentativa** del DB schema

STUDENT

| Name | Student_number | Class | Maj |
|------|----------------|-------|-----|
|------|----------------|-------|-----|

COURSE

| Course_name | Course_number | Cred |
|-------------|---------------|------|
|-------------|---------------|------|

PREREQUISITE

| Course_number | Prerequisite_number |
|---------------|---------------------|
|---------------|---------------------|

SECTION

| Section_identifier | Course_number | Se |
|--------------------|---------------|----|
|--------------------|---------------|----|

GRADE_REPORT

| Student_number | Section_identifier | G |
|----------------|--------------------|---|
|----------------|--------------------|---|

Figura 1. Schema diagram

- **schema construct**: insieme tra **schema e dati** dei DB

- **database state**: **snapshot** in istante t del DB, si definisce quindi ai suoi contenuti
- **valid state**: si definisce funzionante se il suo contenuto **soddisfa i vincoli** per quello schema
- **data dictionary**: insieme per salvare schema e altre info

C. Schema

Possiamo avere 3 **livelli di schema**:

- 1) **interno (fisico)**: come i dati devono essere salvati e come posso accederci
- 2) **concettuale**
- 3) **esterno**: per descrivere le view dell'utente

Per passare da uno schema ad un altro ho bisogno di un **mapping** per capire a cosa corrisponde un elemento. Avremo:

- **logic data independence**: se voglio **cambiare lo schema concettuale** senza cambiare quello fisico
- **physical**: devo **cambiare lo schema fisico** senza cambiare quello concettuale

D. Tipologie di DBMS

Possiamo avere più tipologie di DBMS:

- **centralized**: dove abbiamo tutta l'**elaborazione su un unico nodo**
- **2-tier**: si specializza in termini di server per ogni blocco di funzionalità che devo offrire
- **clients**: per far accedere gli utenti
- **DBMS server**: per eseguire query e transazioni tramite API

IV. DATA MODELING USING THE ENTITY-RELATIONSHIP (ER) MODEL - 28.09.22

Partendo dal mini-world serve **capire i requisiti utili**. Bisognerà far gestire, all'applicazione, alcuni dati per poi visualizzarli (requisiti relazionali).

La procedura sarà:

- 1) acquisizione dei data requirements
- 2) conversione in un modello concettuale
- 3) applicazione dell'algoritmo di mapping
- 4) DBMS si occupa di physics design ed internal schema

in parallelo avremo la **gestione delle transazioni** del mini-world estraendo i functional requirements per effettuare una functional analysis che genera delle transazioni ad alto livello.

A. Entity-Relationship (ER)

Per la scelta degli elementi avremo:

- **entità' (sostantivi)**: **oggetti o cose specifiche** presenti nel mini-world che bisogna rappresentare
- **relazioni (verbi)**: **collegano le entità'**. Il **grado di tipo** della relazione è il **numero di partecipanti a quella relazione**, identificando quante volte la relazione viene percorsa.

Può:

- essere **ricorsiva** se si riferisce ad una stessa entità
- avere un suo attributo definito dall'azione che sta compiendo

- **attributi (proprietà')**: **descrittori** per ogni entità
- **record**: **insieme degli attributi** che si danno ad un entità
- **dato singolo**: ha un unico valore
- **dato composto**: dati da un **insieme di più descrittori**, notazione: $...(, ...)$
- **dato multivalore**: attributi che hanno **n-uple di valori**, notazione: $...$
- **attributo chiave**: **identificare univocamente tutti i record**. Si può usare anche un'unione tra attributo chiave e un altro attributo
- **entità' debole**: entità che **da sola non può esistere**, quindi dipende da un entità più forte. Le sue relationship saranno deboli anche esse. Questa entità **non ha un attributo chiave** ma ha almeno un **attributo in comune con l'entità forte**.

Piccoli **accorgimenti da avere**:

- scritto da **sx a dx** e dall'altro verso il basso
- nomi delle **entità' al singolare**
- **verbi alla terza persona** e attivi o passi per capire da che parte si deve leggere la relazione
- per la **cardinalità** mi chiedo per un solo elemento quante entità potrà avere dell'altro a cui è relazionato. Può essere rappresentata tramite:
 - **vincoli di dipendenza esistenziale**: 1:1, 1:N, M:N dove bisogna mettere la cardinalità nel lato opposto
 - **min max**: dico che posso avere da un min a un max di record che percorrono la relazione dando in vincolo di intervallo (sarà di aiuto a chi farà il database quando dovrà gestire un warning)

I database NoSQL saranno esenti da una modellazione così pesante.

Potremmo incomberci in **relazioni di livello piu' alto** nel caso in cui ci trovassimo a descrivere relazioni con complessità alto. In gnerale **si cerca di evitare e di farlo con relazioni binarie** per evitare complicazioni nell'implementazione.

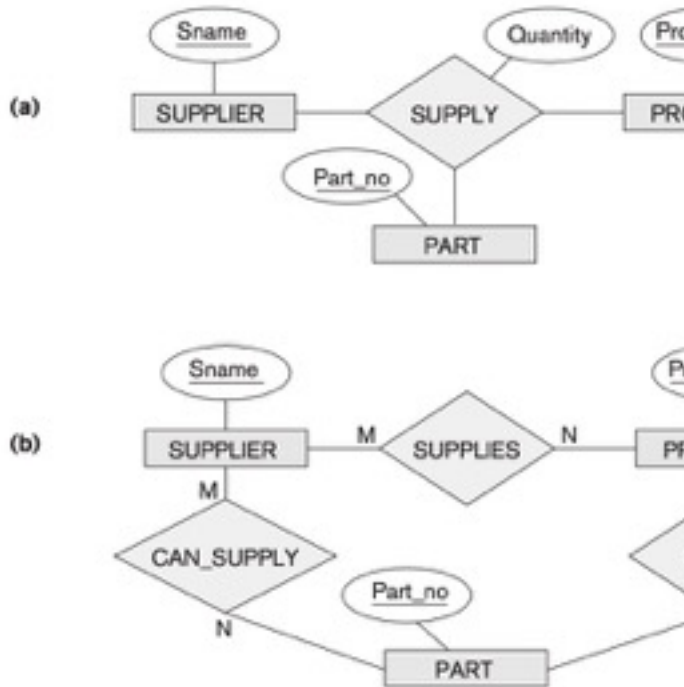


Figura 2. Relazione di livello alto

V. THE ENHANCED ENTITY-RELATIONSHIP (EER) MODEL - 04.10.22

A. Superclassi e sottoclassi

L'idea è di andare a creare una **gerarchia**:

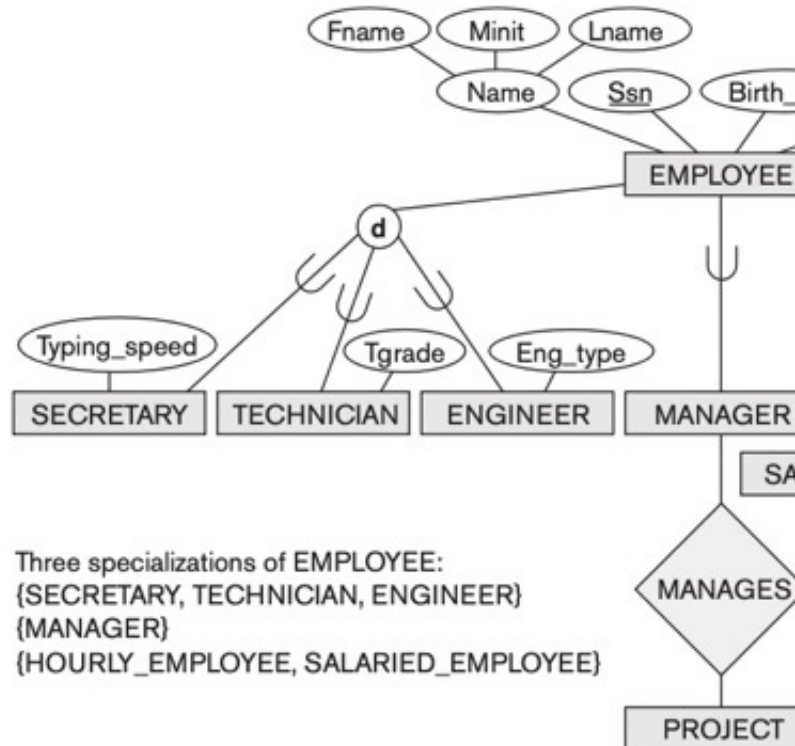


Figura 3. Gerarchia con disjoint

Per modellarlo mi chiedo quali siano le **caratteristiche che hanno in comune alcune entità**, allora tutti gli **attributi in comune vanno nella superclasse**. **Ogni entità DEVE avere i suoi attributi specifici** ma non ho un attributo chiave dato che viene preso dalla superclasse.

B. Graficazione superclassi e sottoclassi

La graficazione avrà per:

- **specializzazione diretta**: si ha un segmento
- **gerarchia (IS-A)**: si ha un segmento con un nodo con:
 - **d -> disjoint**: NON POSSO avere un'entità che è contemporaneamente due o più sottoentità (**solo una**)
 - **o -> overlap**: posso avere un'entità che è contemporaneamente due o più sottoentità (**almeno una**)
 - **U -> union**: **raggruppa** entità di tipo diverso

le quali potranno avere **partecipazioni totali o parziali** che indicano se la superclasse deve o meno scegliere tra le sottoclassi.

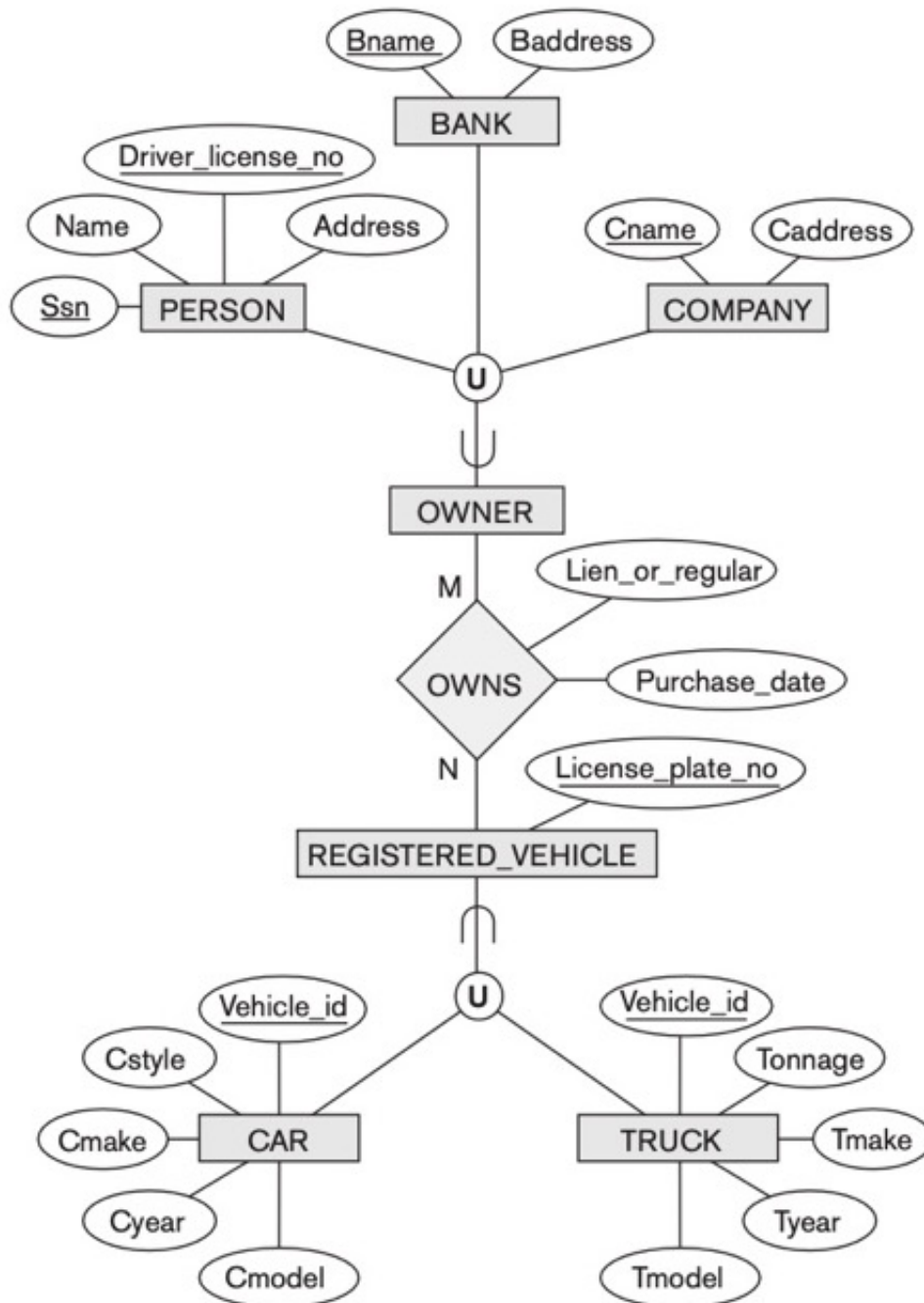


Figura 4. Gerarchia con disjoint

RIFERIMENTI BIBLIOGRAFICI

- [1] <https://en.wikibooks.org/wiki/LaTeX/Hyperlinks>