Appunti BD

La caratteristica principale dell’approccio database è l’uso dell’astrazione sui dati attraverso i **data model**

L’astrazione serve a poter operare su problemi ignorandone aspetti inutili, nel caso dei database ci astraiamo dalla memorizzazione dei dati, utilizzando un modello dei dati = concetto per descrivere una base di dati

* **Modelli**

Un modello rappresenta attraverso l’astrazione una certa realtà, fornendo un insieme di strutture simboliche per descrivere la rappresentazione della realtà, in modo da poter interpretare ed operare sulla realtà

**In sintesi**:

1:Un modello fornisce quindi le regole di rappresentazione per descrivere il **miniworld** (che è la realtà da rappresentare) producendo quindi la rappresentazione di questo, sulla quale poi si può operare e questo rappresenta il **Modello concettuale**, mentre il **Modello Logico** traduce le strutture concettuali in strutture logiche interpretabili dal **DBMS**.

I data model si differenziano in:

* **Data model di alto livello (concettuale)1**
* **Data model rappresentazionali (di interpretazione)** concetti comprensibili all’utente finale ma pur sempre implementabili
* **Data model di basso livello (fisici)** concetti che specificano in modo preciso in che modo vengono memorizzati i dati

**Data model di alto livello**:

vengono utilizzati concetti come entità, attributi e relazioni:

* + Entità: rappresenta un oggetto o concetto del mondo reale (miniworld)
  + Un attributo rappresenta una caratteristica importante di un’entità
  + Una relazione tra 2 o più entità rappresenta un’interazione fra queste

Il più comune data model di alto livello è il modello entità-relazione

**Data model rappresentazionali**:

comprendono i tre data model più usati dai DBSM commerciali:

* + Modello relazionale
  + Modello reticolare
  + Modello gerarchico

Tali modelli rappresentano i dati attraverso dei record (record-based)

* I modelli Object-Oriented si impongono, per la loro astrazione tra rappresentazionale e concettuale

**Data model fisici:**

rappresentano la memorizzazione dei dati nel calcolatore

**Schemi e Istanze**

* Lo schema è la struttura logica del database
* L’istanza è il contenuto del database in un istante di tempo

La loro relazione è come quella di tipi (schemi) e variabili (istanze)

Lo schema del database è la sua descrizione (**metadati**), viene specificato nella progettazione del database e è pensato per essere cambiato frequentemente, per rappresentare lo schema si utilizza un diagramma di schema che mostra la struttura dei record ma non il loro contenuto (astraendosi quindi dal valore dei record); nel DBSM lo schema viene memorizzato nel catalogo per poter essere richiamato ogni volta che è necessario. Ogni oggetto nello schema è detto **costrutto di schema**

Le istanze di database, è anche detto stato del database è l’insieme dei dati contenuti nel database in un particolare istante di tempo.

Dato uno stato del database, ogni costrutto per ogni istante ha un insieme di istanze

Quindi per ogni dato schema di database possono essere costruiti diversi stati del database, e aggiungendo o rimuovendo dati da questo si va a modificare lo stato del database

Alla creazione di un database viene specificato nel DBMS uno schema, inizialmente quindi il DB è “stato vuoto”, prima di essere riempito per la prima volta, per poi passare a “stato iniziale”. Il DBSM, quindi, ha anche la responsabilità di garantire che ogni stato del DB sia valido, ovvero soddisfacendo la struttura e i vincoli dello schema.

**Architettura three-schema:**

* Supporto di viste multiple per gli utenti
* Schema per la memorizzazione nel DB
* Condivisione dei dati ed elaborazione delle transazioni in modalità multiutente

Viene quindi suddiviso il DB in tre livelli:

* **Livello esterno**: **User view** (livello utente), prende i dati dallo schema concettuale per poterli mostrare agli utenti del DB
* **Livello concettuale**: **Schema concettuale** ottiene le informazioni dallo schema interno
* **Livello interno**: **Schema interno** ottiene ed organizza, attraverso il suo schema, i dati dalla memoria: file su dischi

**Livello esterno**:

* Definisce un sottoinsieme del DB su una particolare applicazione
* Include una o più viste utente (schemi esterni)
* Usa un data model di alto livello
* Ogni schema esterno mostra all’utente solo la parte in cui è interessato nascondendo tutto il resto

**Livello concettuale**:

* Descrive la struttura del DB per una comunità di utenti
* Si concentra sui tipi dei dati e sulle entità: i loro tipi, le loro operazioni, relazioni e vincoli, nascondendo tutto il resto come la specifica della memorizzazione dei dati
* Può usare un data model ad alto livello o uno concettuale

**Livello interno**:

* Descrive in modo specifico le specifiche della memorizzazione fisica del DB
* Usa un data model fisico, descrivendo i dettagli completi sul data storage e gli accessi del PATH del BD

**Mapping tra livelli dell’architettura**:

* Ogni livello è solo una descrizione dei dati, in quanto gli unici dati realmente esistenti sono a livello fisico
* In un DBSM con architettura Three-Schema ogni utente o gruppo di utenti utilizza una propria vista esterna
* Il mapping è un processo che trasforma le richieste e i risultati.
* Il DBSM trasforma una richiesta specificata da uno schema esterno, in una richiesta allo schema concettuale secondo i suoi vincoli, e a sua volta in una richiesta allo schema interno secondo i suoi vincoli per poter operare sui dati fisici.

L’utilità dello schema Three-schema è di poter cambiare uno schema ad un particolare livello di un database senza dover operare su tutti gli altri schemi, questa è l’**indipendenza dei dati**

**L’indipendenza può essere**:

* **Logica dei dati**:
  + Lo schema concettuale può essere cambiato senza dover toccare in seguito lo schema esterno, la modifica può avvenire aggiungendo o rimuovendo un data item/record dal database (ampliandolo o riducendolo)
* **Fisica dei dati**:
  + Lo schema interno può essere cambiato senza modificare gli altri schemi (concettuale o esterno), un cambiamento allo schema interno può avvenire per via di un’implementazione migliore o di una riorganizzazione di un file fisico

**VANTAGGI:**

* l’architettura consente di ottenere facilmente una vera indipendenza
* il database risulta più flessibile e scalabile, in quanto può essere aumentato

**SVANTAGGI:**

* l’architettura a più livelli richiede dimensioni maggiori per via delle informazioni su come trasformare le riciheste e i dati tra i vari livelli
* i due livelli di mapping, per via delle traduzioni, creano un’inefficienza nel DBMS

**Linguaggi DBMS**

In un’architettura senza separazioni nette dei livelli viene utilizzato un DDL (data definition language), il cui compilatore ha la funzione di memorizzare la descrizione dello schema del DBSM e di interpretare i DDL statement per identificare la descrizione dei costrutti nello schema

In un’architettura con una netta separazione tra i livelli si usa un SDL(Storage Definition Language) e un DDL per la specificaizone dello schema concettuale, mentre i mapping tra i vari livelli possono essere specificate con entrambi i linguaggi

Nelle Three-Schema si utilizza un VDL (View Definition Language) per specificare le visite degli utenti e i mapping dal livello esterno e il livello concettuale

Il ciclo di vita dei sistemi informativi è l’insieme e sequenzializzazione da analisti e progettisti fino ad utenti nella creazione e l’utilizzo del sistema stesso, è detto ciclo per la sua iteratività

Si articola in:

1. **studio di fattibilità** -> si analizzano le aree di applicazione dal miniworld, studio di costi/benefici, calcolo della complessità dei dati e vengono impostate le priorità tra le applicazioni
2. **raccolta ed analisi dei requisiti** -> definizione delle funzionalità del sistema attraverso una raccolta dei requisiti degli utenti
3. **progettazione** -> progettazione di dati (DB) e funzioni (sui DB)
4. **realizzazione** -> implementazione del sistema informativo, caricando il DB e testando gli stream
5. **validazione e collaudo** -> Verifica che il sistema soddisfi i requisiti desiderati
6. **funzionamento** -> Fase operativa del sistema, in seguito alla validazione di tutte le funzionalità

Ognuna di queste deriva e rimanda al suo superiore per via di informazioni che derivano da quello superiore e informazioni prodotte a quello inferiore

Per garantire una buona qualità del prodotto bisogna seguire una **metodologia di progetto** che ha la proprietà di essere **generale**, **qualità di prodotto in termini di correttezza**, **facilità d’uso delle strategie e dei modelli**

Progettazione concettuale (Analisi = cosa bisogna progettare) -> Progettazione logica -> Progettazione fisica

Sia la progettazione logica che fisica rispondono alla domanda “Come bisogna progettare”

Il frutto delle varie fasi sono schemi di alcuni modelli di dati

Un modello di dati è un insieme di costrutti utilizzati per organizzare i dati di interesse e descriverne la dinamica

Componente fondamentale: costruttori di tipo (meccanismi di strutturazione), analogamente ai linguaggi di programmazione che permettono di costruire nuovi tipi, il modello dei dati prevede alcuni costruttori

**Modelli:**

* **Modelli concettuali** -> permette di astrarsi (quindi allontanarsi da dettagli inutili e avvicinarsi ad una soluzione universale per quella classe di problemi) e di rappresentare i dati indipendentemente dal sistema, usato nelle prime fasi della progettazione (es. modello Entity-Relationship ER)
* **Modelli logici ->** Concretizzazione del modello concettuale in un DBSM realmente esistente, indipendente dalle strutture fisiche. (es. relazionale, reticolare, gerarchico, ad oggetti)

**Indipendentemente dal DBSM:**

Dal miniworld vengono analizzati e ricavati:

* Requisiti funzionali -> analisi funzionale -> specifica transazioni ad alto livello
* Requisiti sui dati -> disegno concettuale -> schema concettuale (datamodel di alto livello

**In base al DBSM:**

* Specifica di transazioni ad alto livello + schema logico -> **disegno programmi applicativi**
* Schema concettuale -> Disegno logico (data model mapping) -> schema logico (usando il data model del DBSM) + specifica transazioni ad alto livello -> **Disegno fisico**
* **Disegno fisico -> schema intero**

**Raccolta dati ed analisi dei requisiti:**

* **Output:** 
  + Requisiti sui dati
  + Requisiti funzionali

**Disegno DB concettuale**:

* Creazione dello schema concettuale con un data model ad alto livello astraendosi dal DBSM
* **Output:**
  + Descrizione dei requisiti degli utenti, con vincoli, relazioni e tipi

**Disegno DB logico**:

* Implementazione del DB concettuale attraverso un DBSM commerciale
* **Output:**
  + Schema del DBSM basato sul modello scelto

**Disegno DB fisico:**

* Specifica delle strutture di memorizzazione, dei path e dell’organizzazione dei file
* Progettazione programmi applicativi e implementazione delle transazioni

Modello ER

Creato da Peter Chen

È un data model ad alto livello

Utilizzato in tools per la progettazione di database

Predecessore dell’EER (Enhanced-ER)

Descrive i dati attraverso tre concetti:

* **Entità**, corrispondono a classi di oggetti, che per via dell’astrazione hanno esistenza autonoma (quindi possono corrispondere con un oggetto fisico ma anche con uno concettuale), vengono identificate da un nome univoco nello schema e graficamente sono rappresentati con un rettangolo con all’interno il nome
* **Attributi,** sono proprietà specifiche delle entità che le caratterizzano, sono identificate anche loro da un nome univoco nello schema e graficamente sono rappresentati da un’ellisse con il nome dell’attributo all’interno
* **Relazioni**

Le entità con attributi di base uguali sono raggruppate in un “tipo di entità”

Nel modello ER esistono poi altre sottocategorie di attributi:

* Semplice o Composto (divisibile in più attributi)
* Single-valued o Multivalued (più di un valore)
* Memorizzato o Derivato (calcolabile)

Gli attributi composti possono formare una gerarchia:

es.

* + Indirizzo:
    - Via
      * Numero
      * Nome
      * Interno
    - Città
    - Stato
    - Codice

Attributi memorizzati e derivabili:

es.

Età = attributo memorizzato -> Data\_di\_nascita = attributo derivabile

o viceversa

Alcuni valori possono anche essere **null** ovvero non noto/mancante/non applicabile

Attributi chiave = è un attributo di un’entità che deve avere un valore univoco per ogni entità

Spesso un insieme di attributi possono costituire una chiave, raggruppandoli insieme componendo un attributo composto che diventa attributo chiave

Graficamente un attributo chiave si rappresenta sottolineando il nome dell’attributo nell’ovale

Es laboratorio (agenzia di viaggi) \*inserire foto\*

Bisogna eseguire ***l’analisi dei requisiti:***

* Analizziamo sostantivi e verbi delle singole frasi:
  + Frase 1:
    - *Agenzia di viaggio (frase 1) rappresenta la realtà di interesse*
  + Frase 2*:* 
    - *ogni* ***gita******ha*** *un* ***codice,*** *un* ***responsabile*** *…, una* ***data di partenza*** *…* ***elenco di partecipanti*** *…* ***fa riferimento*** *ad un* ***itinerario****,* in questo caso: gita, codice, responsabile, data di partenza ed elenco di partecipanti sono tutti sostantivi che possono essere caratterizzati. Inserendo ogni attributo al sostantivo che è caratterizzato da questo.
      * Gita
        + Codice
        + Data di partenza
      * Responsabile
        + Nome
        + Cognome
        + Codice fiscale
        + Recapito telefonico
      * Partecipante
        + Nome
        + Numeri di telefono\*
        + Data di nascita
        + Età
        + Codice fiscale
      * Itinerario
    - Verbi:
      * Ha [Gita, Responsabile]
      * Fa riferimento [Gita, itinerario]
      * Partecipa [Gita, partecipante]
  + Frase 3:
    - *di ogni* ***itinerario*** *si vuole* ***memorizzare****…* ***durata… prezzo … elenco tappe*** *~~corrispondenti alle~~* ***è composto*** *dalle* ***località visitate*** *con* ***durata del soggiorno*** *in ogni località… le* ***tappe*** *sono caratterizzate da un* ***nome univoco*** *e una* ***descrizione***
      * Itinerario
        + Durata
        + Prezzo
      * Tappa/Località visitata
        + Nome univoco
        + Descrizione
    - Verbi:
      * È composto [itinerario, tappa]
        + Durata soggiorno

Bisogna poi verificare che i vari concetti abbiano almeno una caratterizzazione e che siano collegati ad almeno un altro concetto e che non siano a sé stanti, viene poi una fase di arricchimento

Per esempio, a partecipante può essere aggiunto **allergie**\*

tappa

Itinerario

Partecipante

responsabile

È composto

partecipa

ha

segue

gita

La durata soggiorno, per esempio, come l’età possono essere derivate da altri dati memorizzati come la data di nascita per l’età del partecipante e la durata dell’itinerario può essere derivata dalla somma delle varie durate delle singole tappe

**Esempio: Database Company**

Organizzato:

* in **dipartimenti:**
  + Nome
  + Numero
  + Impiegato che lo gestisce
  + {sedi}
* Progetto:
  + Nome
  + Numero
  + Luogo
  + Addetti
    - Impiegato
    - Ore
* Nome e numero sono entrambi attributi chiave sia in progetto che in dipartimento
* Impiegato:
  + Nome
  + Ssn
  + Sesso
  + Indirizzo
  + Stipendio
  + Data di Nascita
  + Dipartimento supervisore
  + Lavora su
    - Progetto
    - Ore
* SSN è un attributo chiave, mentre, nome ed indirizzo sono attributi composti
* Persone a carico:
  + Sesso
  + Data di nascita
  + Impiegato
  + Nome persona a carico
  + Parentela

Relazioni implicite che possono essere specificate in fase di arricchimento, come l’attributo ***manager*** si riferisce comunque ad un ***impiegato*** che gestisce il **dipartimento**

**Relazioni:**

Corrispondono a legami logici tra le entità:

* Tipo di relazione = associazione tra n tipi di entità, tipicamente sono binarie però possono essere anche a più entità correlate da una relazione, e il suo grado è definito dal numero di entità (binaria, ternaria, quaternaria …)
* Occorrenze (istanze di relazione) = associano n entità dei tipi di relazione richiesti
* Ogni tipo di entità è detto partecipare ad una relazione

Sintatticamente una relazione può essere un verbo, come nel caso del db company “lavora per” che mette in collegamento “impiegato” e “dipartimento”, e un’istanza di relazioni è un insieme di relazioni che lega due tipi di entità, che matematicamente sono rappresentate da un prodotto cartesiano, quindi una funzione

Grado di una relazione:

Rappresenta il numero di entità coinvolte nella relazione generalmente le più frequenti sono quelle binarie come appunto quella tra impiegato e dipartimento

Rappresentazione di relazioni:

Viene rappresentato un rombo con un nome identificativo al centro, con due o più linee uscenti collegate alle entità partecipanti alla relazione

**Nomi di ruolo**

Rappresentano letteralmente il ruolo dell’entità partecipante in ciascuna istanza di relazione

Es.:

**Impiegato** ha il ruolo di ***addetto*** o ***impiegato***

**Dipartimento** ha il ruolo di ***dipartimento*** o ***datore di lavoro***

**-Relazioni ricorsive**

Una relazione si dice ricorsiva quando si connette da entrambe le parti dello stesso **tipo di entità**.

Es: impiegato e supervisiona, dove un impiegato ne supervisiona un altro, vanno quindi specificati i nomi di ruolo per rendere comprensibile graficamente la relazione

**-Vincoli sui tipi di relazioni**

Anche questi sono determinati dalla situazione del miniworld, hanno il compito di limitare le combinazioni possibili di entità che possono partecipare alle varie istanze di relazione, insomma si restringe il dominio della funzione

Due tipi di vincoli **strutturali**:

* Rapporti di cardinalità -> restringe il numero di istanze di relazione a cui un’entità può partecipare
* Vincolo di partecipazione -> specifica se l’esistenza di un’entità dipenda da una relazione ad un’altra entità:
  + Partecipazione totale = dipendenza da essa
  + Partecipazione parziale

***Rapporto di cardinalità:***

* Indica il numero minimo e massimo di istanze di relazione a cui le entità vincolate possono partecipare
* Vanno specificati sul collegamento tra entità e relazione con una parentesi tipo “(MIN,MAX)”, dove MIN rappresenta il numero minimo di collegamenti che devono esistere e MAX il massimo, dove una di queste può anche essere un numero N per rappresentare che ne può avere molteplici(senza limite), dove MIN può assumere 0 o 1, dove 0 rappresenta un collegamento opzionale e 1 un collegamento obbligatorio

**Entità debole** = entità la cui esistenza dipende dall’esistenza di un’altra entità

Come attributi presentano:

* Mancano di attributi chiave propri
* L’identificazione avviene attraverso una relazione

Possono essere utili per sostituire attributi composti che rendono la lettura del progetto troppo complessa, si crea quindi un’entità debole che viene identificata attraverso le relazioni con le altre entità (forte) che sono legate a questa

Ridondanza: relazioni che creano un ciclo

**Business rule**

È uno strumento utilizzato per descrivere una proprietà di un’applicazione non rappresentabile per intero nel modello ER

Sono di tre tipi:

* Descrive un concetto rilevante
* Vincolo di integrità sui dati = enunciate attraverso **deve**/**non deve**
* Derivazione = enunciate attraverso **si ottiene** o espressioni simili

Viene espresso attraversi il dizionario dei dati, composto da:

* Tavola delle entità = nome, descrizione generica in linguaggio naturale, elenco degli attributi con descrizioni e possibili identificatori
* Tavola delle relazioni= come la tavola delle entità però con un elenco delle possibili entità coinvolte al posto dei *possibili identificatori*

Le business rule vengono implementate attraverso **clausole SQL** durante la definizione dello schema, oppure attraverso **trigger** oppure attraverso procedure scritte nella fase di applicazione

**Esempio BD**

DB della ASL:

il sistema deve servire per la memorizzazione dei dati sei pazienti dell’asl.

Sostantivi:

* **Un paziente** 
  + è identificato da:
    - Codice tessera sanitaria
  + Caratterizzato da:
    - Nome
    - Cognome
    - Indirizzo
    - Data e luogo di nascita
    - *Allergie*
    - *Gruppo sanguigno*
* **Esame medico**
  + Identificati:
    - Codice numerico
  + Caratterizzati:
    - Descrizione dell’esame
* **Gli ospedali dell’ASL**:
  + Identificati da:
    - Codice numerico
  + Caratterizzati da:
    - Nome
    - Indirizzo
* **I laboratori** che eseguono gli esami
  + Identificati da:
    - Codice univoco **all’interno dell’ospedale dell’ASL**
  + Caratterizzati da:
    - Nome del lab
    - Piano di ubicazione
    - Numero di stanza
* **Per ogni prenotazione** di un esame (da parte di un paziente):
  + Data dell’esame
  + Ora dell’esame
  + Costo del ticket
  + Urgenza dell’esame
  + (Paziente, Laboratorio)
* **Reparto** 
  + Identificati da:
    - Codice numerico univoco **all’interno dell’ospedale**
  + Caratterizzati da:
    - Nome del reparto
    - Numero di telefono
* **Personale del reparto**:
  + Identificato da:
    - Codice fiscale
  + Caratterizzato da:
    - Nome
    - Cognome
    - Indirizzo di domicilio
      * Via
      * Cap
      * Numero civico

Verbi:

* Eseguire (esame, laboratorio)
* Disporre (Ospedale, laboratorio)
* Prenota (Paziente, Prenotazione)
* Effettuare (Prenotazione, Laboratorio)
* Contiene (Ospedale, Reparto)
* Appartiene (Reparto, Personale)

Riguarda

Esame

Paziente

Prenota

Ospedale

Eseguito

Prenotazione

Prenotazione

Contiene

Svolto

Dispone

Appartiene

Reparto

Personale

Laboratorio

Laboratorio

**Modello EER**

Viene creato per andare oltre i limiti dell’ER, ovvero:

* Non possono essere definite relazioni tra un tipo di entità ed un tipo di relazione
* Non possono essere definite relazioni tra un tipo di entità ed una collezione di tipi di identità
* Non è possibile specificare vincoli tra i tipi di relazione
* Mancano i concetti di **generalizzazione** e **specializzazione**

EER aggiunge i concetti di:

* Sottoclasse e superclasse
* Specializzazione e generalizzazione
* Categorie
* Ereditarietà degli attributi

**Sottoclassi e Superclassi**

Un tipo di entità può avere un sottogruppo delle entità che lo compngono in base a determinate caratteristiche, può essere utile per memorizzare informazioni specifiche, per esempio: al tipo di entità employee si possono creare alcune sottoclassi come: segretarie, ingegneri, tecnici, impiegati part-time e altri, e ognuna di queste può avere degli attributi specifici, come nel caso delle segretarie la velocità di digitazione, o per gli impiegati part time la paga oraria, ognuno però è limitato alla propria entità

In questo caso Employee rappresenta una superclasse per le classi di impiegato part time, segretarie e le altre, che a loro volta sono sottoclassi di Employee, il loro legame è quello IS-A (o IS-AN) -> Segretaria IS-AN Employee

Un’entità che fa parte di una sottoclasse appartiene necessariamente anche alla sua superclasse, pur rimanendo la stessa entità, cambia di ruolo, in questo caso prima nel ruolo di Segretaria nella classe ed Employee nella superclasse, anche se l’entità resta la stessa

Per questa caratteristica allora, una sottoclasse deve necessariamente possedere gli stessi attributi della superclasse a cui appartiene, questa rappresenta l’ereditarietà

**Specializzazione e Generalizzazione**

Rappresentano i concetti per determinare sottoclassi e superclassi, rispettivamente il processo associato ai due tipi di classe sono la specializzazione e la generalizzazione, graficamente una specializzazione viene indicata da una linea (singola=specializzazione parziale-> non tutti possono fare parte di quella specializzazione o doppia=specializzazione totale -> possono fare parte tutti di questa specializzazione) che congiunge una superclasse e un cerchietto che può indicare la disgiunzione da cui poi si diramano altri collegamenti alle sottoclassi, la disgiunzione è utile per indicare che non possono sovrapporsi due sottoclassi, infatti o si appartiene ad una o all’altra ma non entrambe

Le sottoclassi hanno l’utilità di poter inserire e rendere più chiari attributi che potrebbero essere forzosi o anche sbagliati da attribuire alla superclasse, che appunto sono specifici per una categoria di entità appartenenti alla superclasse ma non a tutti (processo top-down)

L’utilità di questa organizzazione è poi ridurre i controlli che verrebbero scritti nel DBMS, rendendo così l’applicazione più efficiente evitando controlli inutili

La generalizzazione è un processo astratto inverso alla specializzazione, che consiste nel riconoscere attributi comuni a tutte le entità, così da creare una superclasse con delle eventuali sottoclassi per quanto riguardano altri elementi non comuni a tutti (processi bottom-up)

I vincoli di superclassi e sottoclassi possono essere:

* Disgiunto = una sottoclasse esclude l’altra
  + Totale = doppia linea
  + Parziale = singola linea
* Con Overlap
  + Totale = doppia linea
  + Parziale = singola linea

**Data model relazionale**

È il modello più diffuso a livello teorico e commerciale, i suoi pro sono la sia semplicità, che i suoi solidi fondamenti matematici.

Infatti, si basa sul concetto matematico di **relazione** = insieme di un prodotto cartesiano, che hanno come rappresentazione naturale attraverso delle tabelle, il DB in questo caso è rappresentato come una collezione di relazioni

Siano D1,D2,D3…Dn insiemi, Il prodotto D1xD2xD3x…xDn è l’insieme di tutte le n-uple ordinate tali che, d1 D1, d2 D2, … dn Dn  
il numero di n-uple si dice quindi cardinalità della relazione in quanto una relazione è un sottoinsieme. Es:

D1={a,b} e D2={x,y,z}

D1xD2 ={ (a,x), (a,y), (a,z), (b,x), (b,y), (b,z) }

Ed r una relazione sottoinsieme di D1xD2, r={ (a,x), (a,y), (b,y), (b,z) }

Nel caso del DB ogni dominio della relazione è un attributo, e ad ognuno di questi è associato in nome per identificarli, ogni riga della tabella è detta **tupla**

E in aggiunta, ogni elemento di ogni dominio è atomico, cioè indivisibile (minimizzati al punto che non possono essere semplificati ulteriormente) questo è un **vincolo di dominio**, e un metodo per identificarlo è specificare il tipo di dato che lo compone, un insieme(R) di questi domini identificati da un nome identificativo (A) si dice schema di relazione: R={A1,A2,A3,…An} dove il dominio di A1 è D1 e così via.

Ogni riga (o tupla) r rappresenta un’istanza di relazione del tipo r(R)

L’ordine dei vari domini o delle tuple non conta nella relazione, da ciò una tupla può essere definita come un insieme di coppie (attributo, valore)

Da ciò poi due relazioni possono essere equivalenti anche se l’ordine dei loro attributi e delle loro tuple è diverso

L’insieme degli attributi di sk è detto **superchiave**, dove sk rappresenta una tupla

Mentre una chiave è un insieme di attributi che permette di identificare una tupla, si definisce poi una **chiave primaria**, che deve avere la caratteristica di invariabilità nel tempo, viene quindi scelta una delle **chiavi candidate**, nel caso del DB company si sceglie SSN come chiave primaria, e quindi per la sua definizione ogni tupla che include SSN è una superchiave

Una chiave deve essere **univoca** e non devono esistere più attributi con la stessa chiave **vincolo di chiave**

**Vincolo di integrità di entità** = nessun valore di una chiave può essere “**null**”, poiché non si potrebbe identificare l’entità nella tupla

**Vincolo di integrità referenziale** = il riferimento di un attributo(a1) di una relazione R1 qualsiasi deve riferire necessariamente ad un attributo(a2) esistente di un’altra relazione R2, quindi, le relazioni R1 ed R2 si dicono **relate** attraverso l’attributo a2

Insert è un’operazione sulle relazioni e può violare tutti e quattro i tipi di vincoli, nel caso avvenga per gestirlo si può rifiutare l’inserimento

Delete può violare il vincolo sull’integrità referenziale, per gestirla si può:

* Rigettare la cancellazione
* Propagare la cancellazione
* Modificare i valori dell’attributo referenziante = viene posto a null
* Combinare queste 3 sopra

Modify su una chiave primaria consiste nel cancellare una chiave primaria ed inserirne un’altra

Mentre su una chiave esterna bisogna prima verificare che riferisca ad una tupla esistente

**Prima intercorso su ER ed EER**

**Ristrutturazione e Schema Logico**

Il passo successivo allo schema concettuale è lo **schema logico**, che è in grado di descrivere tutti gli aspetti descritti nello schema ER o EER

Per la traduzione in questo schema bisogna prima *ristrutturare* ottimizzando lo schema e semplificandolo, la ristrutturazione oltre che sullo schema concettuale si deve basare su un modello logico e anche il carico applicativo, ovvero, la dimensione dei dati e le caratteristiche delle operazioni su questi

La ristrutturazione dello schema avviene attraverso:

* Analisi delle ridondanze
* Eliminazione delle generalizzazioni
* Partizionamento/accorpamento di entità e associazioni
* Scelta degli identificatori principali

Per l’analisi delle prestazioni su uno schema ER vanno considerati:

* Costo di un’operazione (#occorrenze di entità e relazioni da visitare per compiere un’operazione)
* Occupazione di memoria

Per studiare questi due fattori bisogna conoscere:

* Volume dei dati
  + # occorrenze entità e relazioni
  + Dimensione di ogni attributo
* Caratteristiche delle operazioni
  + Tipo
    - Interattiva
    - Batch
  + Frequenza (numero medio di operazioni in un tempo t)
  + Dati coinvolti

Tavole di analisi:

* Tavola dei volumi: **concetto tipo volume**
  + Elencati tutti i concetti dello schema
* Tavola delle operazioni: **operazioni tipo frequenza**
  + Riportata la frequenza prevista e il tipo (batch o interattiva) di ogni operazione
* Tavola degli accessi: **concetto costrutto accessi tipo**
  + Per ogni operazione viene elencato il numero di accessi coinvolti e il tipo di accesso (lettura o scrittura, dove **la scrittura è più onerosa della lettura**)

La differenza tra batch ed interattivo è il tempo di risposta, quelle interattive hanno una risposta “immediata”, mentre quelle batch sono operazioni che hanno bisogno di più tempo per via della maggiore quantità di dati da computare

Dallo schema ER ristrutturato poi bisogna tradurre verso un modello logico

Per semplificare lo schema è necessario eseguire un partizionamento, attraverso:

* Decomposizione verticale
  + Suddivisione di un concetto operando sulla sua struttura
* Decomposizione orizzontale:
  + La struttura rimane invariata ma vengono considerate le occorrenze delle entità
* Accorpamento di entità:
  + Si applica a relazioni 1,1 ed è utile quando le operazioni accedono ai dati di entrambe le entità, può contenere valori nulli
* Decomposizione di associazioni

**Seconda intercorso**

**Esercitazione**

**Miniworld = casa editrice**

Sostantivi:

* Autore delle opere
  + Dati anagrafici:
    - Nome
    - Cognome
    - Data\_nascita
    - Luogo\_nascita
  + C.F. = chiave perché è inutile inserire il cf nei dati anagrafici perché è ridondante
  + Numero di opere prodotte\*
* Opera (superclasse)
  + Titolo
  + Edizione
  + Costo
  + Codice Opera
* Cd multimediale (sottoclasse di opera) = specializzazione del concetto di opera
  + Size in MB
  + Durata
* Libri (sottoclasse di opera) = specializzazione del concetto di opera
  + Numero di pagine
* Tipografia
  + Nome
  + Indirizzo

Verbi:

* Realizza (Autore, Opera)
  + Contributo
* Stampa (Libri, Tipografia)

È stampato

Tipografia

CD

Libro

realizza

Opera

Autore

**UML**

È un linguaggio di modellazione, viene utilizzato al posto del diagramma ER, descrive infatti le classi e le relazioni che intercorrono tra loro attraverso il **diagramma delle classi**

Differenze con ER/EER:

* UML è un formalismo più ricco (più costrutti e più formalità) e più vicino al software (rappresenta gli aspetti comportamentali di un’applicazione) rispetto ad ER che si basa su una rappresentazione strutturale e astratta
* In UML non c’è una notazione standard per gli identificatori (in ER basta sottolineare il nome di un attributo per indicare il ruolo di identificatore)
* In UML è possibile aggiungere commenti al diagramma
* In UML è possibile dare un verso di navigazione di una relazione (in ER è sottinteso il verso per convenzione)

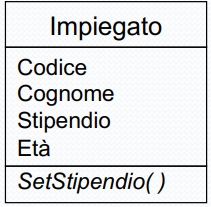
UML si basa su vari diagrammi fondamentali:

* Diagramma delle classi (modello statico)
  + Descrive le caratteristiche statiche e dinamiche delle classi e le associazioni che le legano
* Diagramma degli oggetti
  + Rappresentazioni degli oggetti (istanze delle classi) e i loro collegamenti
* Diagramma dei casi d’uso
  + Rappresenta i metodi e i diritti di accesso
* Diagramma di sequenza
  + Ordinamento temporale dell’invocazione dei metodi tra oggetti
* Diagramma delle attività
  + Descrive il comportamento dinamico di un processo come flussi di attività da svolgere (tipo **flow chart**)
* Diagramma degli stati:
  + Descrive gli stati della vita di un oggetto, attraverso i vari stati che può assumere
* Diagramma di deployement: (modello statico)
  + Descrive l’architettura hardware dell’applicazione

La modellazione strutturale di UML allora si basa su una vista sulla struttura degli oggetti e tutte le loro caratteristiche, questo viene descritto nel diagramma delle classi

**Diagramma delle classi**:

* Classi= componenti principali dei diagrammi delle classi
  + Classi = entità (dell’ER)
  + È possibile aggiungere metodi
  + I dati vengono descritti insieme alle operazioni da svolgere

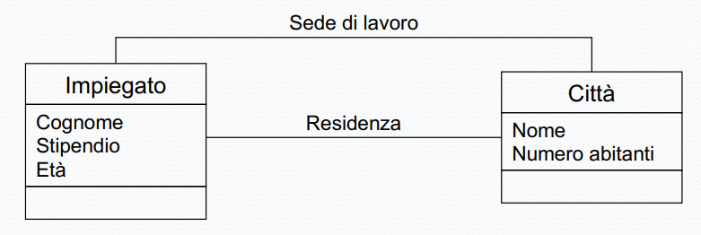
Es di una classe:

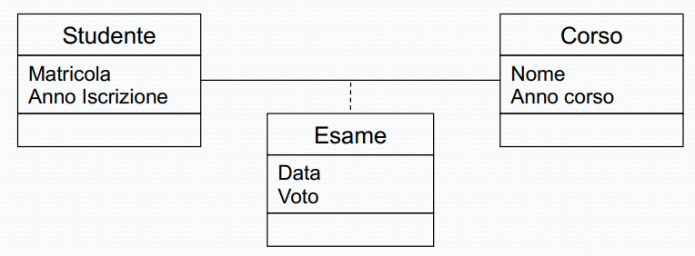
Un attributo in UML rappresenta una proprietà fondamentale degli oggetti di una classe

Non è possibile definire attributi composti, e a differenza di ER è possibile definire il tipo di dato da associare ad ogni attributo

**Associazioni:**

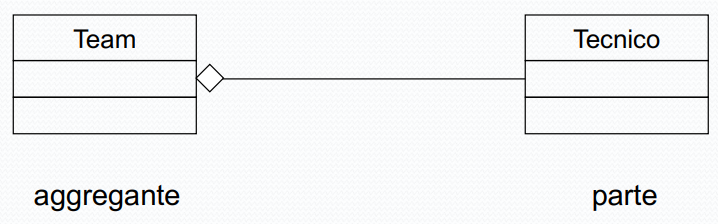
le associazioni (binarie) in UML bisogna semplicemente collegare due classi, spesso anche senza definire il nome della relazione; perciò, è possibile definire più associazioni tra le stesse classi:



Non possono essere associati attributi alle relazioni, bensì si usa una classe di associazione che descrive le proprietà della relazione a cui è collegata, es:

Come per ER, in UML, è possibile definire associazioni n-arie, attraverso un rombo a cui sono collegate tutte le classi

Un’associazione che coinvolge tante classi, attraverso una classe di associazione, si può reificare l’associazione come una classe e legarla alle altre attraverso relazioni binarie.

Le associazioni possono anche indicare l’aggregazione, indicando con un rombo sulla linea che parte dalla classe aggregante alla parte, es:

Il rombo può essere nero o bianco, in base alla relazione tra aggregante e parte, nel caso del rombo bianco la parte può esistere anche al di fuori della relazione con l’aggregante, mentre viceversa con quello nero, la parte ha bisogno della relazione per esistere.

La cardinalità sottintesa (soltanto linea) di un’associazione è (1,1), mentre (0,N) si può identificare con \*, mentre se si ha 1,N lo si rappresenta con 1..\* la rappresentazione della cardinalità poi corrisponde con l’inversione dell’ER, ovvero prendere le 2 massime delle due molteplicità e invertendole

Per l’identificatore interno della classe non esiste una notazione in UML, si possono però definire dei vincoli di integrità su associazioni e attributi specificandoli tra le parentesi graffe, tipo {id} rappresenta che l’attributo è identificatore, mentre assegnare {id} a più attributi indica che si ha un identificatore composto

L’identificatore esterno invece viene indicato con uno *stereotipo*, ovvero <<identificante>> su un’associazione che ha il medesimo ruolo

In UML le generalizzazioni sono indicate con delle frecce dalle sottoclassi alla superclasse, eventuali vincoli vengono rappresentate con le parentesi graffe, tipo {parziale} o {esclusiva}, per indicare se la generalizzazione è con overlap o disgiunta si usano rispettivamente un triangolo nero e uno bianco

In questo caso nel diagramma poi è possibile aggiungere note per la comprensione di vincoli o facilitando la lettura del diagramma stesso.

**Esercitazione (ultima progettazione concettuale)**

Sostantivi:

* Assistito
  + Numero di tesserino sanitario
  + Nome
  + Cognome
  + Indirizzo
* Medici (superclasse)
  + Nome
  + Cognome
  + Indirizzo
  + Codice fiscale
  + Medici a tempo parziale (sottoclasse)
    - Tipo di contratto
  + Medici a tempo pieno (sottoclasse)
    - Data di assunzione
* Strutture ospedaliere
  + Nome univoco
  + Indirizzo
  + Nome

Verbi:

* assistere (medici, assistito)
  + data di inizio
* lavorano (medici, strutture ospedaliere)

**(1,N)** **(0,N)**

lavora

ospedale

Tempo parziale

Tempo pieno

Assistito da

assistito

medico

**M N**

(1,1) **N**

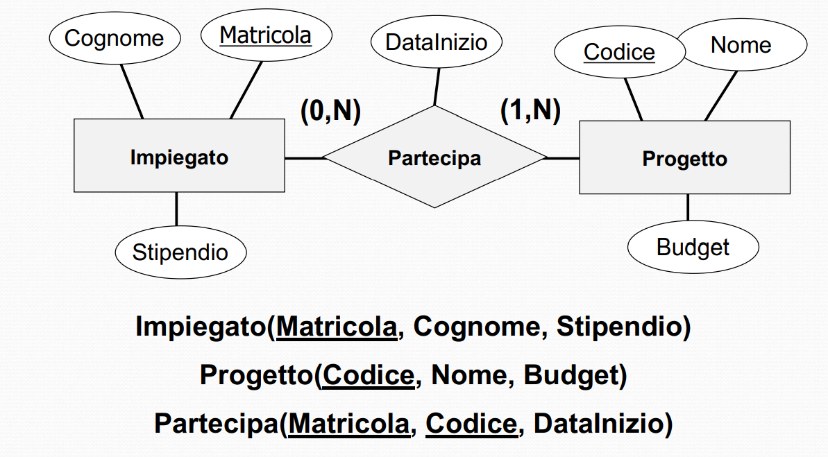
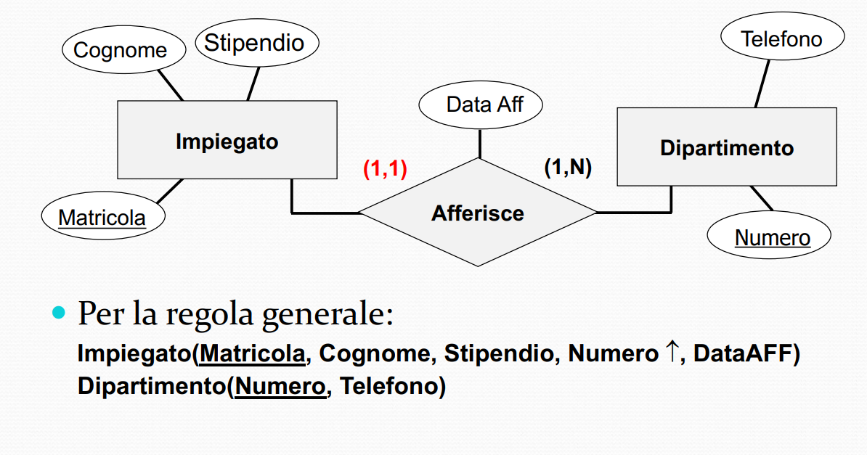
**1**

(1,N)

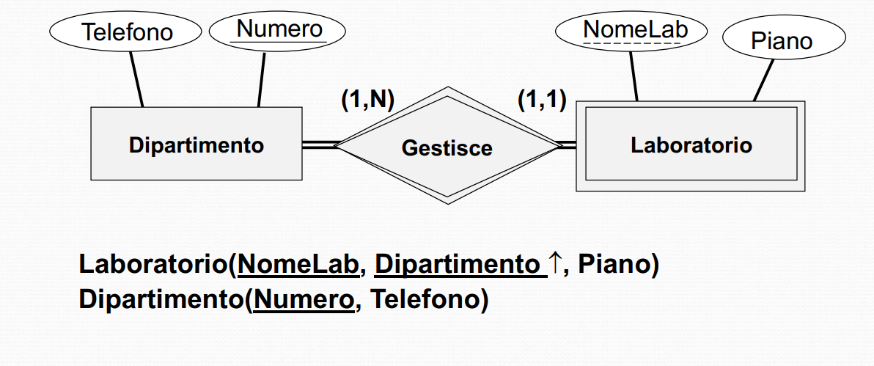
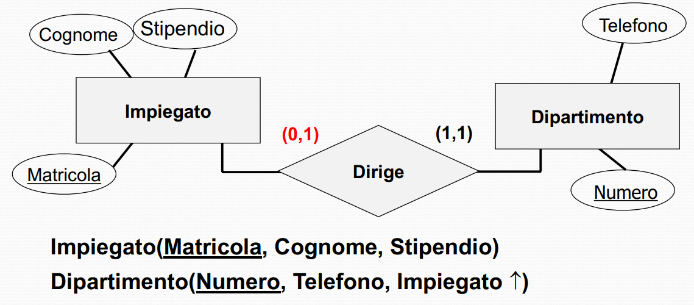
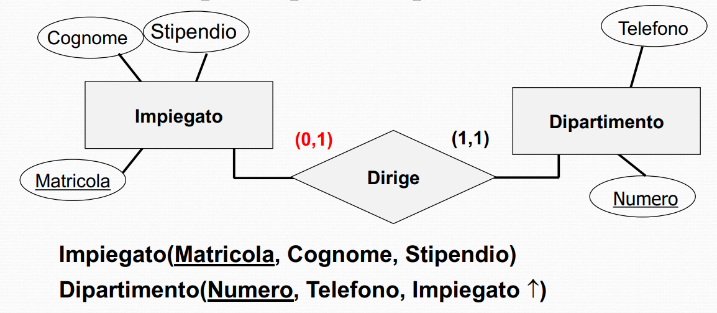
Inizia da qua

**Mapping**

Schematizzazione delle relazioni:

* Nel caso M:N si indicano le entità partecipanti attraverso un vincolo di integrità referenziale, es.
* Nel caso 1:N non viene creata una relazione ma vengono importate la relazione e i suoi attributi dal lato in cui la relazione è 1, es.

Questa è la rappresentazione compatta, conveniente quando il minimo della relazione è 1

* Nel caso di un’entità debole, questa viene mappata indicando come segue un entità come chiave nel mapping di un’altra entità
* Nel caso di una relazione 1:1 si può decidere di inserire un vincolo referenziale nel mapping di una delle due entità.
* Nel caso di una relazione opzionale (0:1) si può inserire un vincolo referenziale nel mapping di un’entità la cui partecipazione alla relazione non è opzionale (1:1 o 1:N).

**La documentazione dello schema logico contiene**

* Risultato della progettazione logica
  + Schema logico
  + Documentazione associata (eliminando alcuni dati della documentazione concettuale)
* Documenti

**Algebra Relazionale**

L’algebra relazionale è un insieme di operazioni possibili per manipolare i dati di un DB, usate per manipolare intere relazioni

L’insieme è chiamato algebra nel puro senso matematico, ovvero:

* È basata su operatori e domini dei valori su cui operare
* Gli operatori mappano gli argomenti da un dominio all’altro
* Un’espressione che comprende operatore e argomenti genera un nuovo elemento nel dominio

Caratteristiche dell’algebra relazionale:

* Dominio: le relazioni
* Operatori sono operazioni interne, ovvero operano su un insieme e restituiscono un suo sottoinsieme
  + Operazioni su insiemi (come in teoria matematica):
    - Unione
    - Intersezione
    - Differenza
    - Prodotto Cartesiano
  + Operazioni specializzate per i DB:
    - Select
    - Project
    - Join
* Il risultato dell’applicazione di un operatore su una relazione è ancora una relazione, D -> D

**Select e Project**

Selezione:

* Decomposizione orizzontale

Proiezione:

* Decomposizione verticale

**Operazione di RENAME**

È un operatore unario utile per rinominare gli attributi risultanti di un insieme risultato di operazioni algebriche

**Operatore Select**: σ

Viene usato per selezionare un sottoinsieme di tuple di una relazione che soddisfano la condizione per la selezione, esempio nel database company:

* σ dno=4 restituisce il sottoinsieme di impiegati che lavora nel dipartimento numero 4
* σ salary>30000 restituisce il sottoinsieme di impiegati che ha un salario maggiore di 30000

Come nelle condizioni si applica la logica booleana, con operatori or and not

Il grado della relazione risultante è uguale a quella della relazione di partenza, e il numero di tuple risultanti si indica con tp dove questa è sempre minore od uguale alla tupla di partenza (tr) mentre è detta selettività il rapporto tr/tp

L’operatore Select è unario ed è commutativo

**Operatore Project: π**

Viene usato per selezionare un sottoinsieme di colonne di una relazione, la sua sintassi è  
π<lista di attributi> (<relazione>) e il grado della relazione risultante è uguale al numero di attributi nella lista indicata e il numero delle tuple risultanti è sempre minore od uguale a quelle originarie

**NON** è commutativa

Operazioni insiemistiche, per renderle possibili devono essere **union compatibili** ovvero avere lo stesso grado e i domini a due a due corrispondenti (devono avere la stessa struttura):

* **Unione**
* **Intersezione**
* **Differenza**

**Operatore prodotto cartesiano**:

Non richiede che le relazioni siano union compatibili, anzi produce tutte le possibili combinazioni tra una relazione R ed una relazione S, se R contiene x tuple ed S contiene y tuple allora R x S contiene x\*y tuple

È un operatore binario

**Operazioni di Join**

Per via della grande frequenza delle operazioni di prodotto cartesiano, combina due tuple relate in una sola tupla, combina un’operazione di prodotto e un’operazione di select in base ad una “join condition”

Una join generica è detta Theta join, mentre la più diffusa è EQUIJOIN che produce sempre una coppia do valori di attributi identici, per evitare queste ripetizioni esiste la Natural Join denotata da \* che evita la inutile ripetizione di più attributi uguali

La cardinalità del risultato dell’operazione di join è compresa tra 0 e il prodotto della cardinalità degli operandi

In algebra relazionale due espressioni si possono dire equivalenti solo se producono lo stesso risultato qualunque sia l’istanza della base di dati

**Operazione di divisione**

È definita come l’operazione inversa del prodotto cartesiano, il risultato è un insieme che estrae dalle tupla del dividendo le altre caratteristiche che **non sono** il divisore, ovvero se R1 ha delle tuple X, Y

Z, H

A, X

ed R2 ha una tupla X, dividendo R1/R2, il risultato Rs è: Y

A

Non sempre uno schema uscito dalla fase di progettazione è privo di difetti, difetti che poi possono portare ad anomalie, la qualità di uno schema può essere stimata in base a due livelli:

* Livello logico
  + Riferimento allo schema di relazione e agli attributi assegnati
* Livello di manipolazione e memorizzazione
  + In base al modo in cui sono memorizzate ed aggiornate le tuple

Informalmente la qualità viene misurati attraverso questi parametri, non da considerare a loro stanti:

1. Semantica degli attributi
2. La quantità di elementi ridondanti nelle tuple
3. La quantità dei valori null
4. La presenza o meno di tuple spurie

**1 - Semantica degli attributi**

La semantica degli attributi indica il modo in cui interpretare i valori di questi ultimi e quindi una semantica più semplice e più sarà facile comprendere e realizzare il disegno dello schema, la semplicità di interpretazione inoltre rende più semplice descriverne le linee guida e rende più facile l’implementazione di queste.

Le **dipendenze funzionali** sono proprietà della semantica degli attributi e esplicitano un vincolo tra insiemi di attributi, o esplicita la dipendenza tra più gruppi di attributi, evitando fraintendimenti.

Forme normali forniscono:

* Un ambito formale per analizzare gli schemi di relazione, in base alle chiavi e alle dipendenze funzionali tra gli attributi
* Una serie di test che vanno eseguite su schemi di relazione individuali, in cui se una schema di relazione fallisce deve essere decomposta in singole relazioni individuali più piccole che passano il test

Infatti il processo di normalizzazione consiste nella decomposizione di schemi di relazione (più grandi) non ottimali in schemi più piccoli al punto di garantire la mancanza di *update anomalies*

**Attributo primo** è un attributo che ricorre in una chiave di uno schema di relazione

Prima forma normale (1NF):

* Gli unici valori consentiti sono **atomici** (indivisibili / singlevalued), questo avviene scomponendo gli attributi multivalued aggiungengo più tuple, per esempio se ad un attributo corrisponde in un a tupla un valore multi valued come segue:
  + - n | {x,y,z}
  + questa diventano 3 tuple:
    - n|x
    - n|y
    - n|z
* questo avviene anche nel caso di un attributo composto, sciogliendo quindi l’attributo composto nei suoi componenti

Seconda forma normale (2NF):

* Si basa sul concetto di dipendenza funzionale piena, ovvero, una dipendenza funzionale che dipende da tutti gli attributi

Terza forma normale (3NF):

* Si basa sul concetto di dipendenza transitiva, ovvero se tra attributi X e Y di una tupla di uno schema di relazione, esiste un insieme Z che non è sottoinsieme di nessuna chiave dello schema (di prima), allora valgono le relazioni X->Z e Z->Y

Forma normale di Boyce Codd:

* Se ogni volta che in una tabella R vale una dipendenza funzionale X -> A, allora X è superchiave di R

**SQL**

SQL (Structured Query Language) è un linguaggio utile per la definizione e manipolazione di un database.

La manipolazione del database avviene attraverso l’inserimento, cancellazione o aggiornamento dei dati del DB. La caratteristica principale di questo linguaggio è la potenzialità delle query.

SQL è un linguaggio dichiarativo, ovvero vengono dichiarate le operazioni, i vincoli e il risultato desiderato dall’operazione

SQL quindi è sia un DDL che un DML

In SQL:

* Tabella significa relazione
* Riga significa tupla
* Colonna significa attributo

Per schema in SQL si intende un insieme di tabelle, domini, viste e altri costrutti come le autorizzazioni

La sintassi per creare uno schema è:

**CREATE SCHEMA** nome\_schema

**AUTHORIZATION** nome utente

Se authorization non viene spsecificato allora lo schema avrà accesso pubblico

Mentre per creare le tabelle si usa:

**CREATE TABLE** nome\_tabella (attributi e vincoli);

es:

**CREATE TABLE** EMPLOYEE

(FNAME VARCHAR(15) **NOT NULL .**

MINIT CHAR .

LNAME VARCHAR(15) **NOT NULL .**

SSN CHAR(9) **NOT NULL .**

… etc …

**PRIMARY KEY** (SSN)

);

Dove varchar corrisponde alle stringhe che hanno come dimensione al più il numero passato tra le parentesi, mentre NOT NULL rappresenta un vincolo al dato che non può essere vuoto(null)

Primary key invece è un vincolo che indica che SSN è la chiave principale

Alter table è una definizione alternativa di una tabella che permette di aggiungere attributi, cambiare la definizione di una colonna e infine anche aggiungere o rimuovere vincoli. Di fatti è quindi una tabella che può essere modificata durante l’esecuzione.

Per la rimozione di un attributo bisogna scegliere la modalità:

* Restrict ovvero rimuove il vincolo solo se non è presente in nessuna relazione o in nessun vincolo
* Cascade elimina la colonna e tutto quello che referenzia o usa quella colonna

Per interrogare un database in SQL si usano:

* **SELECT** <lista\_attributi>
* **FROM** <lista\_tabelle>
* **WHERE** <condizione>