Basi di Dati

DaveRhapsody

4 Marzo 2020

Indice

0.1 Interazione tra campi . . . 4
0.2 Dove si colloca un DB? . . . 4
0.3 Problemi da NON avere . . . 5
0.4 Condivisione dei dati . . . 5

1 DBMS **6**
1.1 Cosa gestiscono . . . 6
1.2 Cosa devono garantire . . . 6
1.3 Concetti fondamentali . . . 6
1.3.1 Schema . . . 6
1.3.2 Istanza . . . 6
1.3.3 Modello . . . 7
1.3.4 Tipo di modello . . . 7
1.4 Architettura dei DBMS . . . 7
1.4.1 Schemi . . . 7
1.4.2 Indipendenza dei dati . . . 7
1.4.3 Le viste . . . 8
1.4.4 Linguaggi del DBMS . . . 8

2 Modello Entità-Relazione **9**
2.1 Concetto di entità . . . 9
2.2 Attributo dell'entità . . . 9
2.3 Le Relazioni . . . 10
2.4 Vincoli di cardinalità . . . 10
2.5 Tipi di assoiazione . . . 11
2.6 Generalizzazioni e Relazioni Is-A . . . 11
2.6.1 Relazione IS-A . . . 11
2.6.2 Generalizzazione tra entità . . . 12
2.7 Attributi composti . . . 13
2.8 Relazioni n-arie . . . 13
2.9 Identificatori/Chiavi . . . 13
2.9.1 Tipi di identificatore/chiave . . . 13

3 Livello logico **15**
3.1 Modello relazionale . . . 15
3.1.1 Definizioni base . . . 15
3.2 Strutture nidificate . . . 16
3.3 Vincoli . . . 17
3.4 Vincolo di integrità . . . 17

3.5 Vincolo di n-upla . . . 17
3.6 Vincolo di chiave . . . 18
3.6.1 Cos'è una chiave . . . 18
3.6.2 Esistenza delle chiavi . . . 18
3.7 Valori nulli . . . 18
3.8 Informazione incompleta . . . 19
3.8.1 Ammissione di valori nulli . . . 19
3.9 Vincoli di integrità referenziale . . . 19

4 Progettazione concettuale **20**
4.1 Necessità di strategie . . . 20
4.2 Definizione di qualità . . . 20
4.2.1 Regole generali per la documentazione descrittiva . . . 22
4.2.2 Organizzazione di termini e concetti . . . 22
4.3 Strategie semplici . . . 22
4.3.1 Strategie di progetto . . . 22
4.4 Strategia Top-down . . . 22
4.5 Strategia Bottom-up . . . 23
4.6 Schema scheletro . . . 23
4.6.1 Strategia Top-down completa . . . 24
4.6.2 Metodologia mista . . . 24

5 Progettazione logica **25**
5.0.1 Efficienza e correttezza . . . 25
5.1 Fasi della progettazione logica . . . 26
5.1.1 Ristrutturazione dello schema ER . . . 26
5.2 Tavola dei volumi . . . 26

6 Lezione WebEx 1 (credo) **28**
6.1 Differenze tra Algebra Relazionale e SQL . . . 28

Introduzione

Un database, o base di dati, o db (che scriverò d'ora in poi) è un insieme di dati, tipo deposito, per qualsiasi genere di uso, sia aziendale, che personale. I suddetti dati sono inseriti, letti, e soprattutto organizzati secondo certe regole.

Alcuni esempi:

- Agende telefoniche
- Studenti di una classe o scuola
- Qualsiasi insieme generico in cui ogni elemento differisce da un altro secondo delle linee guide (campi) che si decidono alla base.

Da Linguaggi di Programmazione sono state viste le Struct, o Record, che sono delle strutture per i dati statiche, concrete ed eterogenee, aventi più campi non necessariamente dello stesso tipo.

Bene, un DB è un array di record, in cui bisogna garantire integrita, consistenza e NON ridondanza dei dati.

0.1 Interazione tra campi

Tra loro i campi di questi record possono interagire, nel senso che partendo dal valore di un determinato campo, si può ricavare il valore di un altro campo. Esempio? Il codice fiscale, che è ricavato da una formula che non ricordo mai nella vita forever MA prendendo come dati il nostro nome, cognome, etc.

Questo sarà un campo calcolato

0.2 Dove si colloca un DB?

- Intefaccia utente
- Applicativi
- Software di ambiente e di sistema
- DB
- Sistema operativo

- Hardware
- Sistema di comunicazione di rete

Impropriamente si può definire nel mezzo

0.3 Problemi da NON avere

- Ridondanza dei dati: non devono esserci dati ripetuti, ogni record è U N I V O C O. Per renderlo tale credo nel corso che vedremo come si fa, spoiler: chiavi, la nostra futura bacinella di bestemmie.
- Rischi di incoerenza: i dati devono essere consistenti, ossia dato un valore, se lo si attribuisce ad un simbolo (dato, variabile, campo), quel valore dovrà essere SEMPRE quello, per ogni volta che si richiama quel simbolo

0.4 Condivisione dei dati

Data un'organizzazione avente più dipendenti, è naturale che la suddetta possa condividere un determinato insieme di dati, infatti ad ogni settore corrisponde un sistema informativo (Per chi ha fatto economia, il SIA).

Cosa accade quando si condivide una risorsa? Esatto, bisogna fare in modo che non avvengano accessi concorrenti, quindi sono implementate funzioni e procedure di prevenzione di questo genere di problemi. Un DB non protetto da modifiche NON consentite, oltre a fare schifo tipo fortissimo forever maonna guarda da bruciarlo, è NON integro.

Un DB deve essere integro

Capitolo 1

DBMS

Il DBMS (Database Management System) è un software in grado di gestire i DB.. E grazie al piffero direi, ma perchè si usa? Perchè un DB è una risorsa condivisa, e siccome potrebbe contenere dati importanti, serve un software che consenta di tenerla pulita e integra.

1.1 Cosa gestiscono

Le moli di dati su cui operano i DBMS sono tendenzialmente di grandi dimensioni, persistenti (con periodo di vita indipendenti dalle singole esecuzioni dei programmi che ci lavorano) e condivise, nel senso che diverse applicazioni ci possono lavorare sopra.

1.2 Cosa devono garantire

Dalle slides si riportano queste 3 qualità:

- Affidabilità
- Sicurezza
- Effiecienza

E tra l'altro bastava anche solo specificare consistenza ed integrità, ma il ci siamo intesi

1.3 Concetti fondamentali

1.3.1 Schema

Lo schema è la descrizione dei campi di una tabella (banalmente, è come una classe in Java)

1.3.2 Istanza

L'istanza è un record allocato a cui assegno un valore per ogni campo (in Java erano gli oggetti)

1.3.3 Modello

Il modello è l'insieme dei vari costrutti (o regole) utilizzati per organizzare i dati e descriverne i cambiamenti nel tempo.

Cosa compone un modello Le strutture di rappresentazione dei dati: Nel nostro caso si analizzerà la relazione. In base alla struttura cambia il modello, ad esempio il modello **relazionale** userà la relazione, mentre il modello a oggetti ad esempio userà altre strutture.

1.3.4 Tipo di modello

Ce ne sono due:

Logico: Vengono utilizzati dai programmi e non dipendono dalle strutture fisiche. Esempio: Relazionale, reticolare, etc.

Concettuale: Sono ancora più in alto del modello logico, quindi sono anche indipendenti dal DBMS, e sono delle descrizioni del mondo reale, usati in fase di progettazione (Quando studieremo il modello E-R partiranno le imprecazioni)

1.4 Architettura dei DBMS

Tra un BD e L'udente (o i programmi) ci sono due schemi, schema fisico e schema logico. Lo schema fisico è vicino al DB mentre il logico all'utente. (Quando verrà detto Utente, si intende sia Umano che Programma).

1.4.1 Schemi

Come menzionato prima ci sono schema fisico e logico, il fisico è banalmente la reppresentazione di files, blocchi di memoria, cache etc. mentre il logico è quello che abbiam visto prima, quindi schema relazionale etc.

1.4.2 Indipendenza dei dati

Il livello logico è indipendente dal fisico, nel senso che se tu hai un record avente dentro un ID e altri due campi, ti importa poco se verrà salvata su un SSD o su un floppy disk di fine 1800, sempre un record dovrai avere.

Osservazione: Prendiamo l'ipotesi di un record avente id e numero di telefono. (Non l'ho specificato ma l'ID è un campo numerico intero che identifica univocamente il record, fine) Se volessimo in modo sadico dividere il numero in prefisso + numero? Esatto, cambia lo schema logico, ed il record passa da 2 a 3 campi.

E lo schema fisico? Dalle slide non si capisce, quindi azzardo una risposta: siccome lo schema logico è indipendente dal fisico, non dovrebbe cambiare.. Ma non ne son sicuro.

1.4.3 Le viste

Se l'utente accedesse di cattiveria direttamente allo schema logico, ad ogni cambiamento allora dovrebbe cambiare tutto il suo programma. Come si risolve? Con le viste (O schemi esterni) che sono dei sottoinsiemi dello schema logico, che quindi contengono quantità di dati limitate (allo scopo di quell'applicativo).

In che senso? Se l'utente è la portineria, gli si crea una vista avente dati legati alla portineria. E sì, schema logico ed esterno sono indipendenti dal fisico, questo significa che l'applicativo non cambierà se cambio il supporto fisico su cui ho messo il DB. E addirittura il livello esterno è indipendente dal logico.

1.4.4 Linguaggi del DBMS

Ce ne sono due, uno è un linguaggio descrittivo dei dati, e l'altro è di manipolazione (sql). Uno descrive le strutture, l'altro ci scrive dentro e legge.

Capitolo 2

Modello Entità-Relazione

Il modello E-R è uno schema concettuale che consente di rappresentare la realtà tramite entità e relazioni tra esse.

2.1 Concetto di entità

E' un'astrazione di un certo insieme di dati (come le classi in Java). Graficamente un'entità è rappresentata da un rettangolo con all'interno il nome dell'entità.

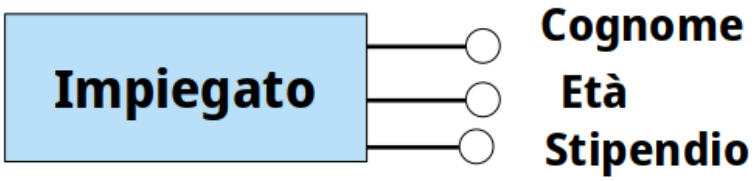
Il nome dev'essere descrittivo , ad ogni entità inoltre occorre dare una definizione. Del tipo Automobile, è quell'oggetto che se è colorato di rosso ed ha un V8 sotto il cofano ti permette di avere orgasmi multipli (dai che mi volete bene, lo so <3).

L'istanza di un'entità è il caso specifico: Automobile = entità, Bmw, Ferrari, Porsche è l'istanza. Il nome che identifica un'entità deve essere singolare ed espressivo, no abbreviazioni, no codici.

2.2 Attributo dell'entità

E' una proprità, o meglio, un campo del record che serve ai fini dell'applicazione , questo valore dipende solo dall'istanza dell'entità, non dipende da altri elementi nello schema. Inoltre ogni attributo ha un intervallo di valori finito. (Il concetto di infinito in informatica non esiste se non per le mie funzioni ricorsive rottissime di Prolog e Lisp)

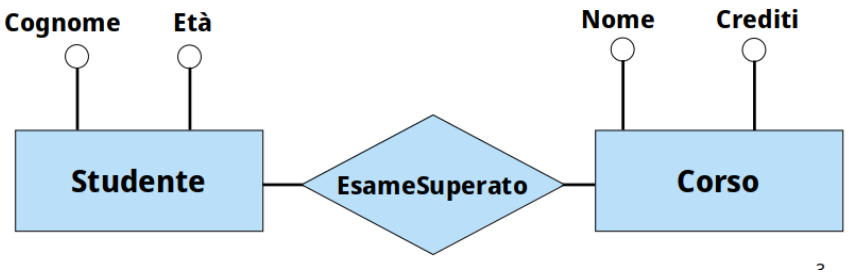
Reppresentazione:



Se il campo è chiave, il pallino diventa pieno. (Sto odiando questa slide, è tutto così non simmetrico che sclero)

2.3 Le Relazioni

Una relazione (o associazione) è un legame logico tra più entità, ed il numero di esse determina il grado (numero di entità obv), si rappresenta con un rombo e dentro ci si scrive cosa lega le entità. Come in questo esempio:



Siccome c'è differenza tra queste relazioni e le relazioni del modello relazionale, per comodità le chiamerò associazioni. Quindi associazioni -> modello e-r, relazioni -> modello relazionale.

Come scrivere le associazioni: Bisogna usare dei sostantivi al singolare. Come da esempio: Se ho due entità studente ed esame ed in mezzo ci metto "Supera" è sbagliato. Si deve mettere "Esame superato". (Alle superiori ero abituato che il verbo andasse all'infinito, che ha più senso ed è più leggibile, ma icchè vi devo dì, noi s'attacca l'asino indoe vuole i' padrone).

Tra due entità posso insierire più associazioni: Ad esempio prendendo uno studente ed un corso, ci puoi applicare due associazioni, una può essere "frequenZa" e l'altra boh.. "Frequenza passata". E da notare che nelle slides ha messo "Frequenta" e "Frequentato in passato".. Niente verbi, ovvio.

Anche un'associazione può avere degli attributi: E' una proprietà **locale** che serve ai fini dell'utente, non è una proprietà della singola entità ma di tutte coloro che sono in legame.

In termini umani: E' una funzione che associa ad ogni istanza di quella associazione un valore che appartiene ad un dominio dell'attributo.

Esempio: Studente - superamento - Esame, un attributo di superamento potrebbe essere Voto. (Tra l'altro lui ha messo esameSuperato.. boh, diobono ok che superato è un aggettivo, ma è participio passato di superare, queste slides sono un casino).

2.4 Vincoli di cardinalità

La cardinalità di una realzione specifica le quantità di interazione di istanze di più entità in un'associazione. si specifica un valore minimo ed uno massimo. Quindi è un dominio, un insieme finito di valori che è possibile assumere.

Definizione: Un vincolo di cardinalità è un limite di istanze di una associazione a cui può partecipre un'istanza di una o più entità, caratterizza meglio il significato di una associazione.

Esempio banale babbarello: Studente - Frequenza - Corso, Da 0 a N studenti frequentano N corsi. 1 corso può essere frequentato da N studenti. 1 studente frequenta N corsi.

2.5 Tipi di assoiazione

Rimanendo nel contesto delle relazioni binarie, ci sono 3 tipi di relazione:

1. Relazione uno a uno
2. Uno a Molti
3. Molti a Molti

Le quali si spiegano da sole, non c'è molto da specificare.

2.6 Generalizzazioni e Relazioni Is-A

Partendo dall'Is-A

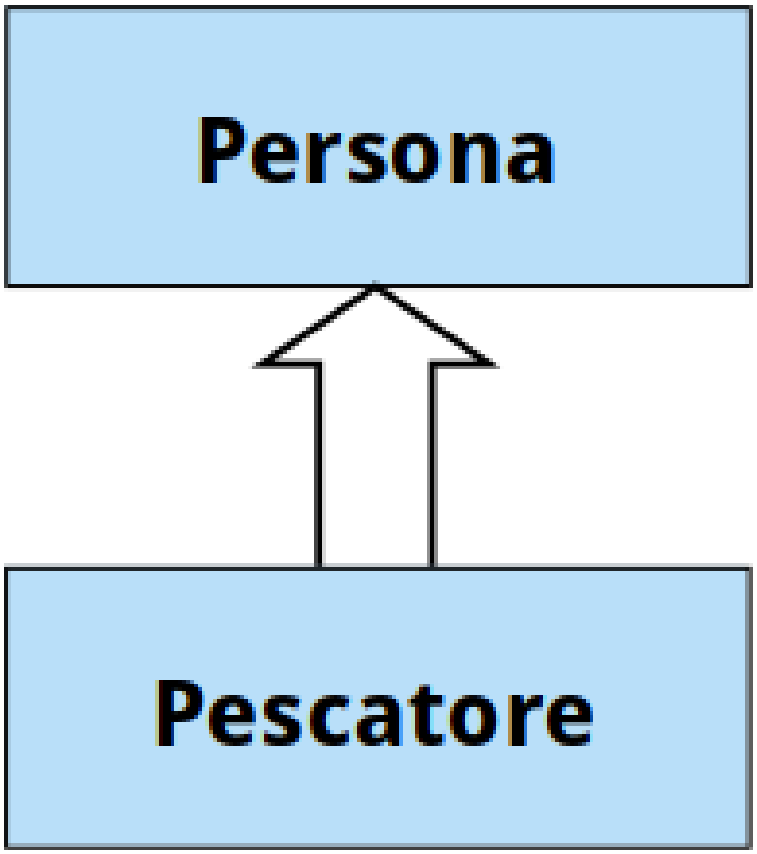
2.6.1 Relazione IS-A

E' una relazione che introduce il fatto che un'istanza possa appartenere a più entità, esempio Pilota, Pilota di Rally. E' anche definita relazione di sottoinsieme, e si utilizza per entità in cui una sia sottoinsieme dell'altra, o meglio. Una deve essere "padre" dell'altra, che si chiamerà "figlia".

Esempio: Fuoristrada Is-A Automobile Is-A Veicolo

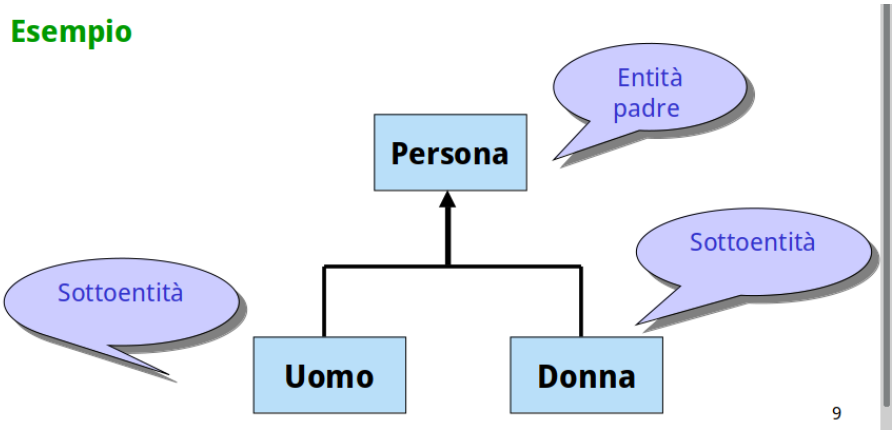
Principio di ereditarietà: Ogni attributo e proprietà dell'entità padre è anche proprietà della sua sottoentità, va da sè che la figlia può avere anche le sue proprietà. (Oh è come le classi di Java, funziona tutto così)

Graficamente:



2.6.2 Generalizzazione tra entità

Hai un'entità padre ed una serie di figlie, quello che fai è con un arco unirle, fine.



Una generalizzazione può essere di due tipi:

- Completa: L'unione delle istanze delle sottoentità è uguale all'insieme delle istanza del padre

- Non completa: in cui questo non accade

Spiegato meglioglioglio: Hai uomo e donna, e poi essere umano, se prendi tutti gli uomini + tutte le donne e fai un insieme contenente tutti essi, ottieni l'essere umano. (Sì, LGBT, abbiate pietà, non è il momento di polemizzare)
Ipotizziamo gli animali: hai cane gatto e criceto, e poi animali.. Beh se sommi tutti i gatti criceti e cani, non ottieni l'insieme degli animali. Mancano i gamberetti per esempio.

Detto scientificamente accurato: Se le sotto entità sono partizioni dell'entità padre, è completa, altrimenti no.

L'entità padre può avere più generalizzazioni

2.7 Attributi composti

Un attributo può esser definito su un dominio di più campi. In altri termini, un attributo può essere a sua volta un record. Tipo indirizzo (composto da via numero e cap), è uno di questi. Potrebbe essere anche un attributo del tipo "dati anagrafici". Quando si ha questa situazione l'attributo è composto.

2.8 Relazioni n-arie

La relazione BI-naria coinvolge due entità, la ternaria tre, la quaternaria.. Via, si è capito, se è n-aria coinvolge n entità. A sua volta anche la relazione può avere i suoi attributi.

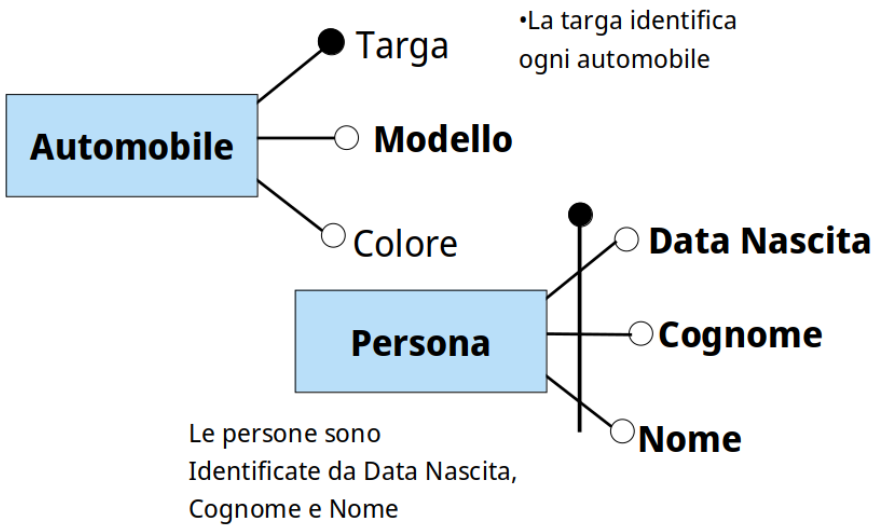
Esistono anche relazioni ricorsive o definite su se stesse: Il classico esempio è il papa. papa - successione - papa, per tenervelo in testa pensate a "Morto un papa, se ne fa n'altro".

2.9 Identificatori/Chiavi

E' un insieme di attributi o relazioni che ti permettono di identificare le istanze di un'entità. (Per chi ha già fatto sta roba sono le chiavi primarie). Ogni entità ha un certo insieme di identificatori (Da 1 a ∞, basta che ce ne sia uno $\forall$ entità)

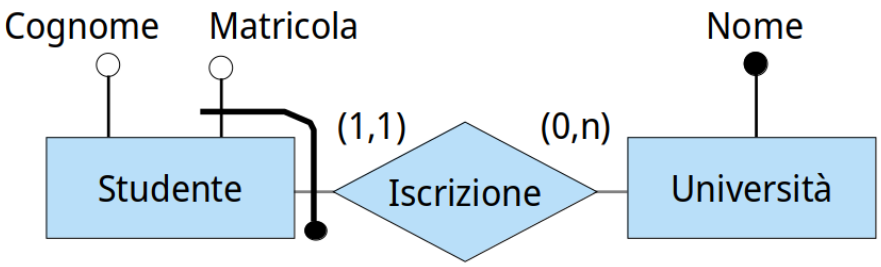
2.9.1 Tipi di identificatore/chiave

- **Interno**: Formati solo da attributi dell'entità stessa. può essere solo uno, possono essere anche più di uno. Come in questo esempio:



- **Esterno**: E' formato da attributi dell'entità e da relazioni che la coinvolgono, oppure solo da queste relazioni.

Esempio grafico:



Capitolo 3

Livello logico

Il livello logico è quello in cui figurano le strutture di rappresentazione logica dei dati. In altri termini, dove c'è uno schema logico che descrive una realtà di interesse. E già qua occhio, non è uno schema concettuale come l'E-R, qua non si parla di generalizzazione della realtà, ma di schema logico dei dati.

3.1 Modello relazionale

Il concetto di relazione è da considerarsi in 3 modi diversi:

1. Relationship (Associazione, classe di fatti)
2. Relazione matematica (Qualunque sottoinsieme del prodotto cartesiano blablabla)
3. Relazione secondo il modello relazionale

(Non mi soffermerò sulla spiegazione della relazione matematica, tenete solo a mente che le n-uple al loro interno sono orninate e che tra le n-uple non c'è ordinamento. Se volet buttateci dentro anche che ogni n-upla è univoca)

3.1.1 Definizioni base

- **Struttura non posizionale**: Per ogni dominio si associa un attributo che ne descriva il ruolo
- **Relazione (o Tabelle)**: Letteralmente la rappresentazione matematica di una relazione in cui i valori per ogni colonna sono dello stesso tipo, e le righe sono diverse. In termini leggibili? Un cazzo di array di record, con tutte le conseguenze che ne derivano.

  Osservazione: Si è specificato "valori" per ogni colonna, NON a caso, perchè nel relazionale ci si basa su valori, ed i riferimenti tra i dati in relazioni diverse sono rapprentati con valori dei domini che compaiono nelle n-uple.

  La domanda è: Per quale f* motivo? Le ragioni sono 3:

  1. L'utente finale vede gli stessi dati dei programmatori, costituiti da tabelle

2. L'indipendenza della rappresentazione delle strutture fisiche, che cambiano in modo indipendente, quindi potete usare i floppyni del Nintendo o un SSD Nvme, I N D I P E N D E N Z A.
3. i dati sono trasportabili da un sistema ad un altro per la ragione del punto 2

- **Schema di relazione**:

$$nomeRelazione(Attributo_1, Attributo_2, Attributo_n)$$

  Facciamo che da ora nomeRelazione $\rightarrow$ nR e Attributo $\rightarrow$ A

- **Schema di Base di Dati**:

$$superRelazione = \{nR_1(A_1, nR_2(A_2, ..., nR_n(A_n,)\}$$

- **N-upla**: Funzione che associa ad un certo insieme X di attributi, $\forall$ attributo A $\in$ X uno ed un solo valore del dominio di A
- **Istanza di relazione**: Data nR(A), l'istanza è l'insieme r di n-uple su nR
- **Istanza di Base di Dati**: Stesso identico concetto ma applicato allo schema della base di dati. Dato lo schema di prima (superRelazione), l'istanza è l'insieme delle relazioni r
- **Valori nulli**: Quando un VALORE non c'è in una determinata colonna, viene considerao Null

3.2 Strutture nidificate

La realtà è rappresentabile con infiniti schemi equivalenti, nel **contenuto informativo**. Ciò vuol dire che schemi equivalenti forniscono lo stesso insieme di informazioni.

Premessa: immaginatevi di volere rappresentare tutti gli scontrini di un ristorante in cui ogni ricevuta ha come campi: {Numero ricevuta, data emissione e le varie voci di costo}

Le voci di costo: Per ogni voce c'è una corrispondenza con un gruppo di portate, e per ciascun gruppo, il numero di porrtate e il costo del gruppo di portate, infine il totale.
Con un semplice schema ad elenco si ottiene alla fine la suddivisione in sezioni, sottosezioni etc., riscrivendolo così vien fuori:

1. Numero ricevuta
2. Data Emissione
3. Voci di costo
   (a) Gruppo di portate
      i. Numero di portate
      ii. Costo del gruppo di portate
4. Totale speso

3.3 Vincoli

Può accadere che dei DB siano sintatticamente corretti MA non rappresentano degli stati realmente possibili nella realtà, ad esempio il 27 e lode (esempio delle slides).

Risoluzione: Per risolvere questo inconveniente occorre aggiungere un vincolo che dica SE è 30 allora può avere lode, altrimenti non può. (Lode può essere benissimo un campo booleano).

3.4 Vincolo di integrità

E' una proprietà che hanno i DB che va soddisfatta da tutte le istanze di uno schema, le quali rappresentano informazioni corrette, consistenti, integre per l'applicazione.

Banalmente: E' una funzione booleana (O predicato for those who love Prolog) che per ogni istanza r dà:

- **TRUE**: se l'istanza rappresenta la realtà
- **FALSE**: In caso contrario

A che servono? Permettono di rappresentare in modo più preciso la realtà, e quindi contribuiscono ad avere dati più consistenti. Inoltre sono utili per progettare perchè lo schema risulta qualitativamente migliore

Tipi di vincolo

Vincolo intrarelazionale: Definito all'interno della relazione. Sono vincoli applicati agli **attributi** (si definiscono anche vincoli di **dominio**), oppure vincoli di n-upla, o comunque relativi all'insieme di n-uple di una relazione $\rightarrow$ Vincoli di relazione

Interrelazionale: Definiti tra più relazioni

3.5 Vincolo di n-upla

Esprime condizioni sui valori di ciascuna n-upla in una relazione, ad esempio non puoi avere due giocatori in una squadra con stesso numero di maglia.

Sintassi

- Esressioni booleane (AND, OR, NOT)
- Atomi da confrontare (>,<, >= etc.)

Per calcolare il valore di verità di un vincolo in una n-upla, occorre sostituire i valori degli attributi nella n-upla, e poi calcolare il valore vero o falso.

3.6 Vincolo di chiave

C'è bisogno di individuare le informazioni che permettono di rappresentare ogni oggetto di interesse con una n-upla differente e identificarlo nel caso in cui ne abbia necessità.

3.6.1 Cos'è una chiave

E' un insieme di attributi che identificano in modo univoco le n-uple di una relazione. Un insieme K di attributi è **superchiave** per una relazione r se r non contiene due n-uple distinte t_1 e t_2 con $t_1[K] = t_2[K]$.
Una chiave tipica è il numero di matricola, che identifica uno studente, e siccome è un solo campo, è anche minimale.
Se invece hai [Matricola + cognome] non sono una chiave MA sono superchiave, e non è minimale siccome c'è già matricola.

Vincoli, schemi e istanze I vincoli corrispondono a proprietà del mondo reale modellato dal DB. Sono proprietà dello schema, fanno riferimento ad ogni istanza. Se hai dei vincoli in uno schema, se la istanza lo soddisfa, è corretto (per ogni vincolo).

3.6.2 Esistenza delle chiavi

Una relazione non può contenere n-uple distinte ma uguali. Ogni relazione ha come superchiave l'insieme degli attributi su cui è definita e ha (almeno) una chiave. Rappresentiamo per ogni dipartimento di un'università i fornitori di beni ed i beni forniti.

Precisazione: Distinte ma uguali significa che hai un'istanza doppia, duplicata, ripetuta, ridondante.

Importanza delle chiavi Identificano e distinguono gli oggetti gli uni dagli altri, e quindi ogni dato diventa accessibile, inoltre permettono di collegare i dati in realzioni diverse.

3.7 Valori nulli

Un valore nullo indica una generica informazione incompleta, assente, non esistente. Un'informazione che a tutti gli effetti non c'è. Occhio a questa cosa, spesso si fa confusione con il concetto di 0 ed il concetto di nullo: Nullo è NON avente valore, mentre 0 è un valore.

Esempio su un conto corrente: la casella che indica quanti soldi sono presenti

$$\begin{cases} SE\ var = null\ allora\ NON\ si\ ha\ questo\ dato \\ SE\ var = 0\ allora\ hai\ 0 \end{cases}$$

Ci sono 3 principali tipi diversi di valore nullo.

Sconosciuto: Quando il valore esiste ma non è noto

Inesistente: Quando invece questo valore non esiste e non è applicabile.

Nessuna informazione: Nel caso in cui o è sconosciuto o inesistente..

3.8 Informazione incompleta

Come si rappresenta a questo punto l'informazione nulla? Teniamo a mente che i dati vanno separati dalle applicazioni, e soprattutto il significato dei dati non deve essere nascosto nei programmi.

Soluzione: Il dominio dei valori si estende aggiungendo un valore nuovo che si chiama NULL, che per l'appunto indica esattamente il valore "inesistente" di prima. Fine

3.8.1 Ammissione di valori nulli

Ci sono casi in cui non si può accettare un valore nullo, poichè renderebbe il record incompleto, non univoco, e quindi ridondanze, inconsistenze etc.

In presenza di valori nulli: I valori della chiave NON permettono di svolgere le due funzioni della chiave (Identificare i record, esprimere i riferimenti ad altre tabelle)

Se la chiave primaria fosse nulla? Verrebbe un agente lì nel punto in cui c'è il programmatore che ha permesso una cosa simile, e gli sparerebbe in fronte.

3.9 Vincoli di integrità referenziale

E' la proprietà per cui le informazioni con stesso significato in relazioni diverse sono correlate, rispettando la proprietà per cui:

- Non è possibile che un oggetto dela realtà
    1. Sia rappresentato nella sua relazione con un altro oggetto
    2. Non sia rappresentato e identificato nella tabella che lo descrive nativamente

Per definizione: Un vincolo di integrità referenziale fra un insieme X di attributi di una relazione R_1 ed un'altra relazione R_2 impone ai valori su X in R_1 di comparire come valori della chiave primaria di R_2

Ok, e se viene violato? In genere i DBMS per evitare problemi di questo tipo agiscono in tre modi principali.

1. Rifiutano l'operazione
2. Applicano una Eliminazione in cascata: Questa è la più cattiva, perchè va a pescare ovunque c'è quel valore chiave, e cancella quel record
3. Introducono dei valori nulli

E se il vincolo è su più attributi? Come in molti altri ambiti è necessario che venga specificato un ordine tra gli attributi.

Capitolo 4

Progettazione concettuale

La progettazione concettuale è la fase che segue la stesura dei requisiti utente, in quesa fase si raccolgono i requisiti e a partire da questi si crea lo schema concettuale (Rappresentazione astratta dei requisiti).

Tipici problemi legati alla stesura dei requisiti: Il cliente non è in grado di spiegarsi, vuole una cosa, ma ti racconta una cosa divesa, ha poche idee, riesce a confondere pure quelle. (Se non sbaglio c'è su Aps una foto carina a tema)

4.1 Necessità di strategie

Quando i requisiti sono semplici, costituiti da poche frasi, allora si può generare uno schema "tutto in una volta", ma se ti becchi requisiti articolati (solito esempio di una banca), serve metodo.

Ad un problema esistono infinite soluzioni, ed infinite soluzioni sbagliate. Pensate quindi cosa vuol dire effettivamente capire se la propria è la strategia giusta.

Per definizione: Si definisce Analisi dei requisiti e della progettazione concettuale, l'insieme di:

- Acquisizione dei requisiti
- Analisi dei requisiti per rimuovere ambiguità e ridondanze
- Costruzione dello schema concettuale
- Costruzione di un glossario dei termini utilizzati

4.2 Definizione di qualità

Una qualità di uno schema concettuale è una caratteristica che sarebbe auspicabile fare avere allo schema, e va garantita alla fine della fase di progettazione concettuale.

Di che qualità si parla?

- Correttezza
  - Rispetto ai requisiti: I requisiti rappresentati nello schema devono essere rappresentati con il significato dei concetti nell'E-R
  - Rispetto al modello: I concetti del modello sono usati nello schema in accordo al loro significato
- Completezza: Tutti i requisiti devono essere rappresentati nello schema
- Pertinenza: Nello schema devono esserci solo concetti che compaiono nei requisiti
- Minimalità: Non ci devono essere più concetti che rappresentano lo stesso requisito.

  Specificazione: In taluni casi può anche essere accettata la non minimalità (o anche ridondanza). L'importante è che sia riconosciuta, in modo tale che a lovello logico sia possibile decidere se mantenere o eliminare la ridondanza.
- Leggibilità: Lo schema deve rappresentare i requisiti in modo comprensibile
  - Grafica: Leggibilità estetica, nel senso che non ci devono essere incroci tra linee, e le line NON devono essere oblique, solo verticali od orizzontali
  - Concettuale: Sarebbe la scelta di strutture del modello che danno luogo ad uno schema più compatto, più semplice da capire.

Come si raccolgono i requisiti? L'analista non ha presente come funziona il dominio d'applicazione, c'è bisogno di individuare diverse fonti informative, a partire dalle quali raccogliere i requisiti

Da dove si attingono le informazioni?

- Dagli utenti tramite interviste o documentazione fornita

  Problemi legati alle interviste: Utenti diversi possono fornire informazioni diverse e usare termini diversi per indicare la stessa cosa.
- Tramite modulistica usata per acquisire i dati

Ok, e che documentazione esiste?

- Normative (Leggi, regolamenti di settore)
- Regolamenti interni, procedure aziendali
- Ralizzazioni preesistenti (I vecchi Db che l'azienda aveva, per intenderci)

4.2.1 Regole generali per la documentazione descrittiva

Per cominciare bisogna scegliere un corretto livello di astrazione, del tipo è meglio usare "animale" o "cane"? Il concetto è capire quanto salire in alto come livello di astrazione. E' utile anche

- Standardizzare la struttura dell efrasi
- Separare le frasi sui dati da quelle sulle operazioni che aggiornano i dati

4.2.2 Organizzazione di termini e concetti

In generale si usa:

- Costruire un glossario dei termini
- Individuare omonimi e sinonimi e fare un'unificazione dei termini
- Rendere esplicito il riferimento tra termini
- Riorganizzare le frasi per concetti

C'è da fare una distinzione tra strategie semplici (Quelle che danno indicazioni per scegliere tra attributo, entità, relazione, relazione is_a e generalizzazione), oppure complesse (Quando si esprime un'idea complessiva su come percorrere il processo di progettazione)

4.3 Strategie semplici

Sono le strategie viste finora per quanto riguardi le lezioni sul modello E-R

- Prima le entità, e attributi di entità
- Poi le relazioni e attributi delle relazioni
- Generalizzazioni

E per la scelta dei singoli concetti

4.3.1 Strategie di progetto

Le solite:

- Top-down
- Bottom-up
- Inside-out: Da un concetto "a macchia d'olio" verso tutti gli atri
- Mista tra tutte queste tre

4.4 Strategia Top-down

Nel top down si possono disciplinare i raffinamenti, che sono sostituzioni di concetti con altri concetti più dettagliati, usando un numero limitato di **primitive di raffinamento**

Le principali sono:

1. Entità → Entità + Attributi
2. Entità → Relazione tra Entità
3. Entità → Generalizzazione tra Entità

4.5 Strategia Bottom-up

Parti da tanti piccoli sottoschemi fatti di concetti, che vai a unire formando uno schema completo, ci sono casi in cui si può attuare una strategia bottom-up o una top-down indifferentemente.

Vantaggi e svantaggi dei vari modelli:

Metodologia	Vantaggi	Svantaggi
Top down	Non ci sono mai ristrutturazioni da effettuare, e in ogni momento il progettista rappresenta sempre l'intero insieme dei requisiti	Poco naturale, richiede una elevata capacita' di dominare specifiche complesse.
Bottom up	Naturale, permette di dividere il lavoro di progettazione tra diversi gruppi	Richiede ristrutturazioni, perche' la progettazione separata di parti di schemi porta a possibili ridondanze e inconsistenze

Metodologia	Vantaggi	Svantaggi
Inside outside	Naturale, perche' parte dai concetti piu' importanti e poi procede verso quelli di dettaglio.	Piu' moderatamente della bottom up puo' portare a ristrutturazioni

A questo punto risulta scontato che la soluzione migliore sia la **mista** per via del fatto che prende i lati positivi di ognuna delle strategie esistenti.

4.6 Schema scheletro

Lo schema scheletro raccoglie i concetti più importanti, che vngono organizzati in uno schema concettuale. Ci sono due main strategie miste, che si differenziano su come si procede dopo aver individuato lo schema scheletro:

1. La prima è tutta top-down: definisce in maggior dettaglio la strategia top-down
2. La seconda è mista, nel senso che prevede passi ispirati alle altre strategie

4.6.1 Strategia Top-down completa

Analisi dei requisiti:

1. Si analizzano i requisiti e si eliminano le ambiguità
2. Si raggruppano i requisiti in insiemi omogenei

Passo base:

1. Si definisce uno schema scheletro con i concetti più rilevanti

Passo iterativo: Si ripete finchè non sono stati rappresentati tutti i requisiti

1. Si raffinano i cncetti presenti sulla base dell eloro specifiche
2. Si aggiungono concetti per descrivere specifiche non descritte
3. Si verifica la qualità dello schema e eventualmente lo si modifica per raggiunggere le qualità desiderate

4.6.2 Metodologia mista

Analisi dei requisiti:

1. Si crea lo schema scheletro

Decomposizione:

1. Si decompongono i requisiti con riferimenti ai concetti nello schema scheletro
2. Si creano i diversi sottoschemi

Passo iterativo:

1. ìSi integrano i vari sottoschemi in uno schema complessivo facendo riferimento allo schema scheletro.

Analisi di quaità

Mista	Coniuga tutti i vantaggi delle precedenti, mantenendo sempre una rappresentazione completa delle specifiche.	Piu' moderatamente della top down richiede una visione complessiva delle specifiche.

Capitolo 5

Progettazione logica

La progettazione logica si pone l'obbiettivo di tradurre lo schema concettuale in uno schema logico che sia in grado di rappresentare lo schema per mezzo di un **modello logico**.

5.0.1 Efficienza e correttezza

Prima di tutto, lo schema nel modello relazionale deve rappresentare la stessa realtà dello schema E-R, e soprattutto le varie query e transizioni devono essere eseguite sullo schema con un costo di risorse ridotto al minimo.

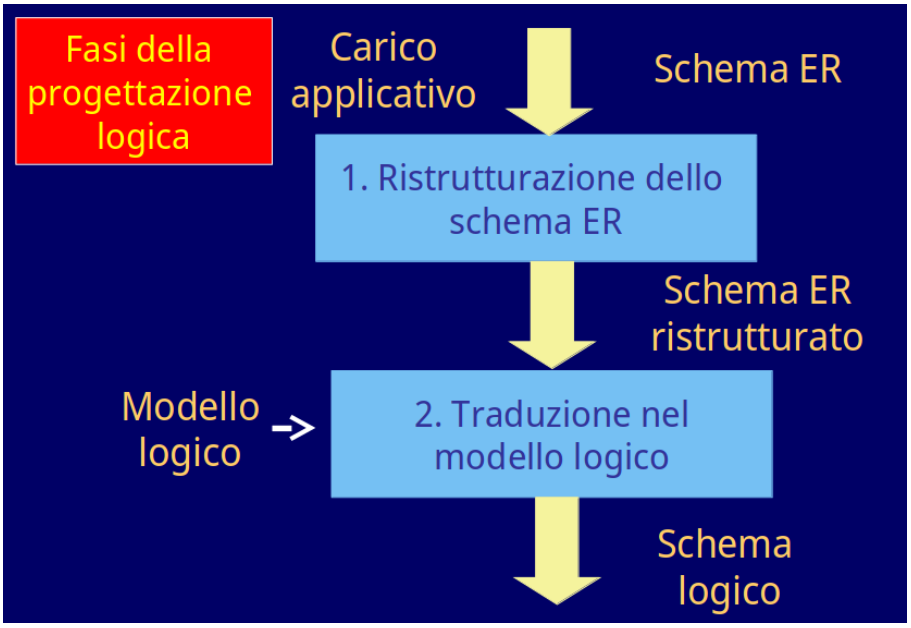
Precisazioni sulla correttezza: Come già detto, lo schema ER e il relazionale devono rappresentare la stessa realtà, devono avere un **contenuto informativo** equivalente. Ci sono però alcune strutture che non sono direttamente rappresentabili:

1. Le generalizzazioni
2. Le associazioni (in particolari le molti a molti)
3. Gli attributi multivalore
4. Le chiavi composte

Precisazioni sull'efficienza: La riduzione del carico applicativo comprende **interrogazioni** (query) e **transizioni**, banalmente operazioni che tirano fuori informazioni, e operazioni che ci scrivono roba.

Fissato il carico applicativo: dobbiamo scegliere quel particolare schema logico che permette di ottimizzare l'esecuzione del carico applicativo.

5.1 Fasi della progettazione logica



E qua comincia ciò che maggiormente ho odiato nel mio 5 anno delle superiori:

5.1.1 Ristrutturazione dello schema ER

La domanda è: Perchè? Perchè bisogna rendere semplice la traduzione nel modello relazionale, e quindi prendete lo schema ER e ne create un altro equivalente (Quindi, che cazzo abbiam creato a fare il primo in principio? Bella domanda). Ah sì poi un altro motivo è l'ottimizzazione del carico applicativo.

Bisogna concentrarsi sul tempo di esecuzioni delle query e sullo spazio in memoria che l'esecuzione di una query occupa.

Esempio che mi è capitato personalmente: Prendete due tabellone gigantescone con dei datoni gigantesconi dentro..ni. Bene, dobbiamo andare a selezionare UN campo di UN record che però si ottieme facendo una join. Ecco, ogni volta crashava tutto perchè questo caricava in memoria T U T T A la tabella del prodotto cartesiano.

Che risorse si hanno a disposizione? **Spazio**: Tavola dei volumi, che per intenderci descrive il numero di istanze delle entità e delle relazioni, e **Tempo**: Tavola degli accessi, che descrive per ogni operazione rilevante, il numero di istanze di entità e relazioni visitate dalla operazione.

5.2 Tavola dei volumi

La tavola dei volumi serve per capire quanto spazio occuperà una base di dati: è una tabella gigante in cui si ha per ogni entità o relazione il numero orientativo di istanze nel corso della vita del DB. La tabella sarà formata da 3 colonne x N righe, dove N dipende da numero di entità+numero relazioni.

Nome	Tipo	Volume
Entità 1	E	α
Entità 2	E	β
Relazione	R	γ

Capitolo 6

Lezione WebEx 1 (credo)

6.1 Differenze tra Algebra Relazionale e SQL

1. L'algebra relazionale è un linguaggio formale, l'SQL si usa quando si lavora sui DB
2. L'algebra relazionale è procedurale, si specifica l'algoritmo, mentre SQL invece è parzialmente dichiarativo, specifica solo il risultato da ottenere, senza specifica degli algoritmi
3. Istruzioni equivalenti differiscono per efficienza, mentre in SQL è solo un questione di leggibilità
4. Nell'algebra relazionale le relazioni sono matematiche, in SQL sono tabelle
5. Negli insiemi non ci possono essere elementi uguali, mentre in SQL sì, possono esserci righe uguali