

Basi di dati

Marini Mattia

1° semestre 2° anno

Indice

1	Introduzione	1
2	Progettazione di una base di dati	3
2.1	Concetti base modellazione logica	3
2.2	Vicoli di relazione	5
3	Riassuntone	8
3.1	Termini	8
3.2	Vincoli	8
3.3	Mapping	9
3.4	Algebra relazionale	9
3.5	Normalizzazione	10
3.6	Livello fisico	10
3.7	Joins	12

1 Introduzione

Per immagazzinare dati abbiamo bisogno di:

- Base di dati raccolta organizzata di informazioni
- Database Managment System o DBMS: pacchetto software per gestire basi di dati

L'insieme di base di dati e DBMS è detto database system

Un DBMS deve fornire come minimo:

- Data Definition Language (DDL) ossia un linguaggio che permetta di definire la struttura dei dati
- Data Manipulationn Language (DML) o quey language, che permetta di modificare ed estrarre i dati
- Garantire robustezza dei dati, rendendo possibile il recupero in caso di corruzione
- Gestire problemi di concorrenza:
 - Evitare interazioni indesiderate tra utenti *isolation*
 - Evitare modifiche incomplete dei dati *atomicity*

Vantaggio database

Utilizzare un database comporta molti vantaggi. Uno dei più importanti è che permette la separazione tra programmi e dati

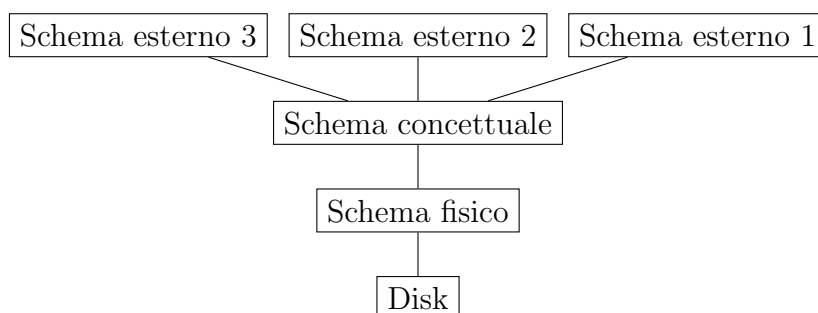
Devo modificare lo schema fisico secondo il quale salvo i dati

Database	Salvataggio su file
<ul style="list-style-type: none">• Modifico database, ma modalità logiche secondo le quali accedo e modifico dati rimangono le stesse• Applicazione <u>non deve essere modificata</u>	<ul style="list-style-type: none">• La struttura del file è memorizzata nel programma• Cambio struttura file• Applicazione <u>deve essere modificata</u>

Dunque il salvataggio su file rende manutenzione difficile e costosa

Livelli di astrazione

- Schema fisico: fornisce dettagli su come i dati sono salvati fisicamente nella memoria
- Schema concettuale: modello di alto livello che rappresenta i dati nei termini del modello del DBMS
- Schema esterno: permette di creare una o più viste per interfacciarsi con il database



Attori sulla scena

- Database administrator: monitora utilizzo DB, regola accesso, acquista risorse hardware e monitora efficienza
- Database designer: crea il design del DB, definendone lo schema e i vincoli sulle transazioni

Utenti finali (utilizzatori database):

- Casual: "accedono da esperti", ma occasionalmente, quando necessario
- Naive
 - Utilizzano funzioni predefinite
 - Es. cassi di banca, impiegati, utenti di siti web...

- Sophisticated: business analysts, data scientists, ingegneri. Utilizzano software molto vicini a quello della base di dati
- Stand-alone: utenti che si creano una base di dati "per conto loro", ad esempio un commercialista che si crea un database con i dati dei suoi clienti

2 Progettazione di una base di dati

Le fasi della progettazione di una base di dati sono le seguenti:

- Fasi principali
 - Analisi dei requisiti
 - Progettazione concettuale
 - Progettazione logica
- Fasi seconda parte
 - Raffinamento dello schema logico (normalizzazione)
 - Progettazione del livello fisico
 - Progettazione applicativa e sicurezza

2.1 Concetti base modellazione logica

Entità

- Un'entità è un oggetto singolo del mondo reale. Ad esempio Mario
- Un entity set è un insieme di entità, quindi ad esempio il gruppo di amici Mario, Paolo e Nicola
- Un entity type è il tipo generico di entità, ad esempio Persona

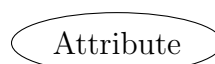
Un'entity è rappresentata tramite un quadrato nei diagrammi:



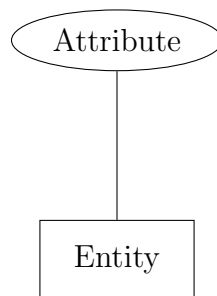
Attributi

- Un attribute è una proprietà usata per descrivere un'entity. Ad esempio, mario avrà codice fiscale, numero telefono...
- Un value set o data type è la variabile che contiene il valore dell'attributo

Gli attributi possono essere semplici, composti e multi valore. Noi useremo quelli semplici e basta. Possiamo ricreare gli altri tramite diversi workaround Un'entity è rappresentata tramite un ovale nei diagrammi:

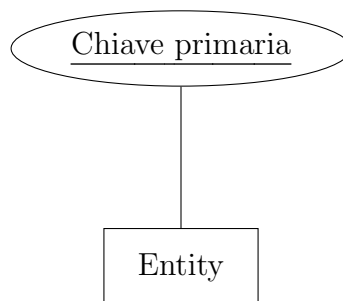


L'attributo viene collegato all'entity type al quale fa riferimento



Attributo chiave

Un attributo chiave è un attributo che identifica in maniera univoca l'entità alla quale fa riferimento. Ad esempio, l'isbn di un libro. La chiave primaria va sempre sottolineata:

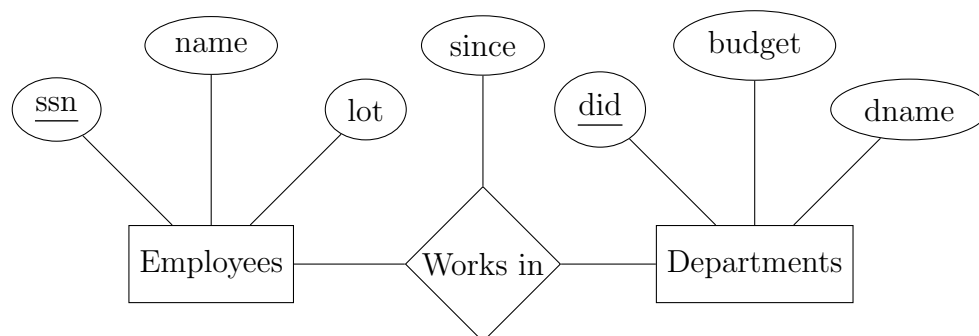


Una chiave può essere inoltre

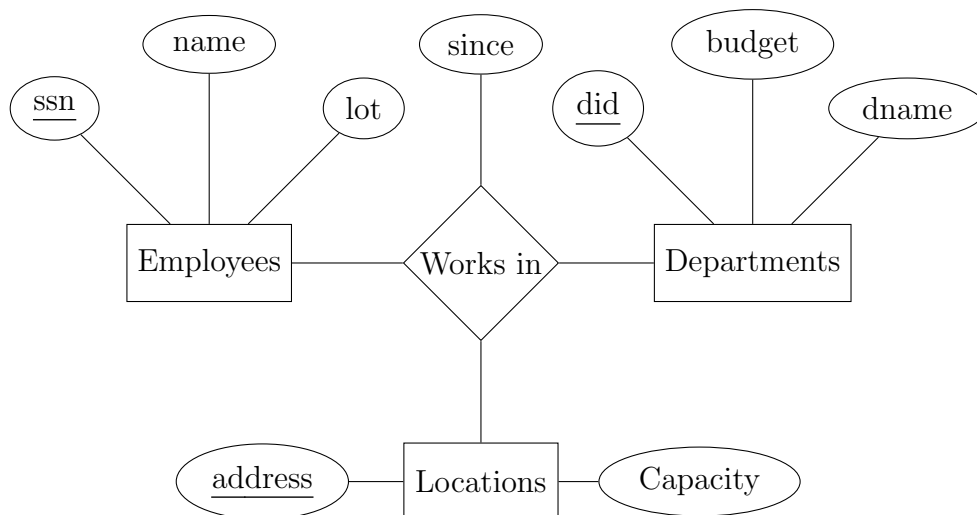
- Composta, ossia formata da più attributi. Ad esempio, nome e matricola
- Multipla, ossia vi sono più chiavi candidate, ad esempio la targa e il numero di telaio di un'automobile. Va sempre comunque indicata la chiave principale

Relazioni

Una relazione è un'associazione tra due o più entity



Una relazione può essere anche a 3 (uwu)



2.2 Vicoli di relazione

Spesso abbiamo bisogno di specificare qualche vincolo riguardo ad una relationship tra due entity

Vincolo di chiave

Si rappresenta con una freccia:



In questo caso significa che ogni dipartimento può avere **al più** un manager

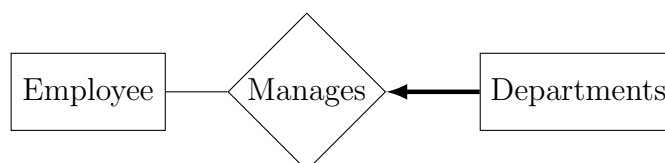
Vincolo di partecipazione

Si rappresenta con una linea grossa:



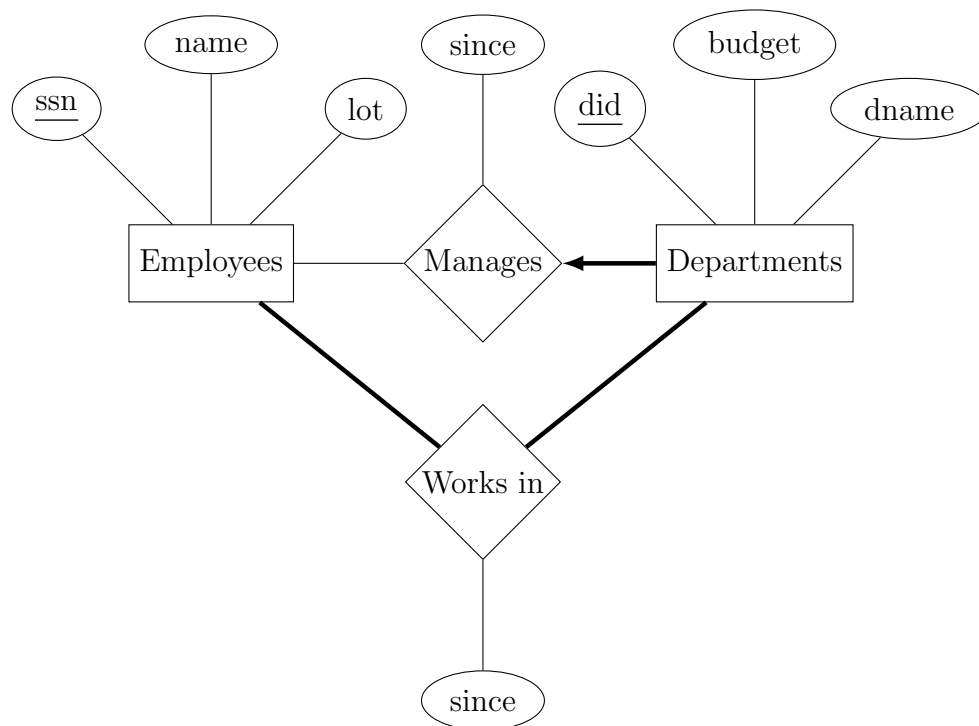
In questo caso significa che ogni dipartimento deve avere **almeno** un manager

I vincoli di chiave e di partecipazione possono essere uniti, come segue:



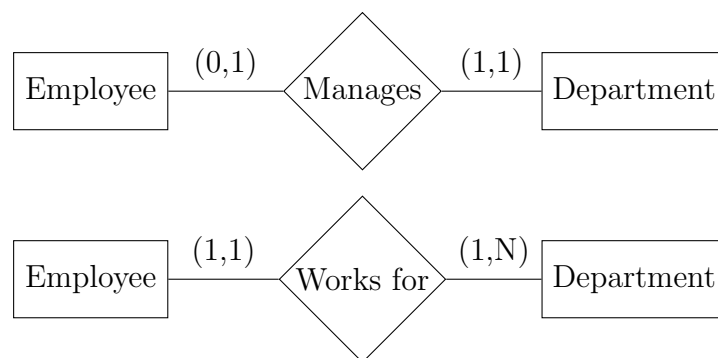
In questo caso significa che ogni dipartimento ha **esattamente** uno e uno solo manager

Esempio completo



Notazione (min, max)

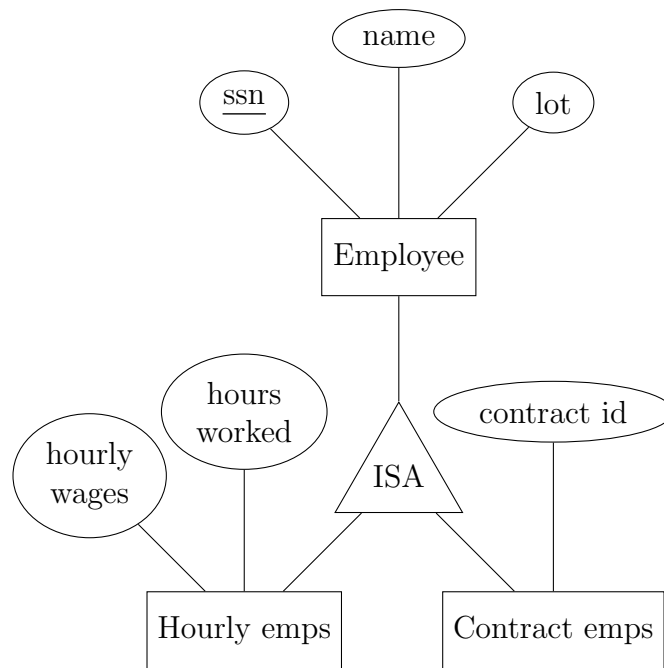
Una notazione alternativa per esprimere dei vincoli sul numero di entity coinvolte nella relazione è utilizzare un disegno come segue:



Nel primo caso si legge: un Employee può essere manager di 0 o 1 department. Un department può essere gestito da un solo manager

Gerarchie di classi

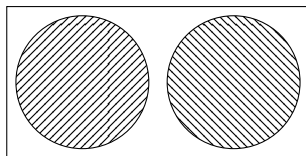
Come usiamo la generalizzazioe all'interno dei class diagrams, possiamo utilizzarla nei diagrammi er



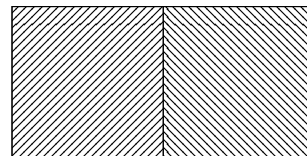
In questo caso si dice che:

- Hourly emps e Contract emps sono generalizzati in Employee
- Employee si specializza con Contract emps e Hourly emps

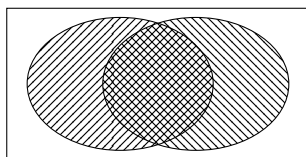
Inoltre possiamo rappresentare i seguenti concetti di overlap e copertura secondo i seguenti schemi grafici



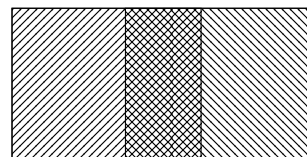
No overlap, copertura parziale



No overlap, copertura totale



Overlap, copertura parziale



Overlap, copertura totale

Aggregazione

3 Riassuntone

3.1 Termini

- Superchiave: un insieme di attributi che identifica univocamente una tupla
- Superchiave minimale: superchiave che se si toglie uno qualsiasi degli attributi non è più chiave
- Chiavi candidate: tutte le superchiavi minimali di una relazione
- Attributo primot: attributo che appartiene almeno ad una chiave candidata
- Attributo non primot: attributo che non appartiene a nessuna chiave candidata
- Schema (scheme): insieme di tabelle e vincoli di integrità di un db

3.2 Vincoli

- Dominio: il tipo inserito in una colonna deve rispettare il formato della colonna stessa:
 - Data type *string, int,...*
 - Range (*int in range 0-120*)
 - Format (*mail, contains "@"*)
 - Enum values (*value can be either a,b,c,...*)
 - Null constraint (*Item cannot be null*)
- Chiave: deve esistere una chiave per ogni relazione
- Integrità delle entità: nessun valore della chiave primaria può essere null
- Integrità referenziale: una foreign key non può far riferimento ad una tupla non presente nella relazione referenziata
- Vincoli di integrità semantica: vincoli che non possono essere espressi dal dbms stesso (*es, tutti gli impiegati non possono lavorare più di 46 ore a settimana*)

Violazione dei vincoli

- Insert: viola tutti:
 - Dominio (valore con tipo sbagliato)
 - Integrità entità (chiave null)
 - Integrità referenziale (Fk non esiste)
 - Integrità semantica
- Delete
- Integrità referenziale (toglie valore referenziato)
- Update: viola tutti:
 - E' come delete + insert

Mantenere integrità

- Restrict: impedisce operazione
- Cascade: rimuove tuple che referenziano tupla eliminata
- Set null/default: setta la chiave referenziante a nulla/default value

3.3 Mapping

- Vincolo di chiave (E_1, E_2 , vincolo su E_2)
 - Opzione 1: $FK(E_1, E_2), PK(E_2)$
 - Opzione 2: aggiungo colonna a E_2 che referencia $PK(E_1)$
- Vincolo di partecipazione totale (E_1, E_2 , vincolo su E_2): non può essere rappresentato completamente, ma solo in parte
 - Aggiungo *not null* sulla $FK(E_1)$ in E_2
- Weak entities (E_1, E_2 , E_2 è weak)
 - $FK(E_2)$ è *not null* (attributo non esiste senza istanza padrone)
 - ON DELETE CASCADE per eliminare weak instance quando viene eliminata istanza padrone
- Gerarchie IS-A (E, C_1, C_2):
 - Opzione 1:
 - * $PK(E) = \text{ssn}$ $PK(C_1) = FK(C_1) = \text{ssn}$ e stessa cosa per C_2
 - * ON DELETE CASCADE: se elimino superclasse elimino anche classe child corrispondente
 - Opzione 2 (se ce copertura totale): elimino classe padre
- Aggregazioni: Relazione esterna ha attributi della relazione interna e la chiave di ciò a cui è collegata. Si può semplificare se
 - Relazione esterna non ha attributi
 - Aggregazione ha vincolo di partecipazione totale su relazione esterna

3.4 Algebra relazionale

- σ : select
- π : project
- ρ : rename
- Divisione: $A(x, y) / B(y)$
 - Raggruppamento per colonne x di A
 - Metto in tabella risultato gli x che hanno tutti gli y di A

$$A/B = \{x \text{ t.c. } \exists \langle x, y \rangle \forall y \in B\}$$

3.5 Normalizzazione

- **Prima forma normale:**

- No attributi multivalore
- No tuple duplicate (esiste chiave)

- **Seconda forma normale:** no dipendenze parziali

- Controlla che tutti gli attributi non primi non dipenda parzialmente da alcuna chiave candidata

Terza forma normale: no dipendenze transitive. Per ogni dipendenza funzionale, $X \rightarrow Y$

Ogni freccia entrante in un attributo non primo deve dipendere da una chiave candidata

- X è superchiave
- Y è primo

- **Forma di Boyce Codd**

- Per ogni df $X \rightarrow Y$, X deve essere una superchiave

3.6 Livello fisico

Terminologia

- P : dimensione della pagina
- S : dimensione del file
- t_R dimensione di una tupla di una tabella R
- P_R : numero di pagine che contengono tuple di R
- $|R.A|$: numero di valori diversi per l'attributo A
- $|R|$: numero di tuple di R
- $|R_{c==c}|$: numero di tuple per le quali è soddisfatta la condizione $c == c$
- $f = \frac{1}{|R.A|}$: selectivity factor

Heap list

- Scan

$$P_R$$

- Equality search

$$P_R$$

- Insert

2

leggo ultima pagina in $\frac{S}{P} - 1$ e eventualmente ne aggiungo una nuova

- Delete

$$P_R + |R_{c=c}| \quad \text{dove } |R_{c=c}| \approx \frac{|R|}{|R.A|}$$

Sorted file

- Scan:

$$P_R$$

- Equality search o ricerca per intervallo:

$$\log_2(P_R) + \left\lceil \frac{|R_{c=c}|}{\frac{P}{t_R}} \right\rceil$$

- Delete:

$$\log_2(P_R) + 2 \cdot \left\lceil \frac{|R_{c=c}|}{\frac{P}{t_R}} \right\rceil$$

il $\cdot 2$ è dato dal fatto che le pagine contenenti le tuple vanno prima lette e poi eliminate

- Insert

$$\log_2(P_R) + 1$$

Se non c'è spazio ho bisogno di spostare tutte le P_R tuple nel caso peggiore

B+ tree

Costo lookup:

$$\log_B(R.A)$$

Equality search unclustered (nel caso peggiore ogni tupla sta in una pagina diversa):

$$\log_B(R.A) + R_{a=a}$$

Equality search clustered (tuple sono contigue quindi complessità diminuisce):

$$\log_B(R.A) + \left\lceil \frac{|R_{a=a}|}{\frac{P}{t_r}} \right\rceil$$

Insert:

$$\log_B(R.A) + 2$$

Hash table

Costo lookup:

$$L_h \approx 1.2$$

Costo insert:

$$L_h + 2$$

3.7 Joins

Nested Loop Join

$$P_R \cdot (P_S + 1)$$

Sort Merge Join

Costo se tuple già ordinate:

$$P_R + P_S$$

Costo ordinamento tuple:

$$2 \cdot P_R \cdot \left(\left\lceil \log_{b-1} \left\lceil \frac{P_R}{B} \right\rceil \right\rceil + 1 \right)$$

Hash join

Costo se tuple già in bucket:

$$P_R + P_S$$

Costo inserimento in bucket:

$$P_R + 2 \cdot |R|$$

Indexed Nested Loop Join

$$P_R + |R| \cdot \text{CostoEqualitySearch}$$