Fondamenti di Basi di Dati

Modellazione di una base di dati e strumenti di gestione

prof. Giovanni Borga

Modellazione di banche dati territoriali

Le tecniche e gli strumenti di modellazione dei dati (territoriali e non) consentono di organizzare per mezzo dei computer informazioni sul mondo reale in modo da poterle elaborare in modo efficiente.
L'obiettivo finale è **ottenere una struttura informativa** costituita da **entità** e relativi **attributi** di interesse, e **relazioni** fra di esse.

Essere in grado di modellare a livello informatico la realtà, implica uno sforzo per:

a) **Capirne gli aspetti** principali
b) Mettere i diversi aspetti **in relazione tra loro.**

E' dunque un'attività fortemente propedeutica alla gran parte delle fasi della filiera di utilizzo della conoscenza per la progettazione, dall'analisi, alla rappresentazione, comunicazione e interazione con il quadro delle conoscenze da parte di attori diversi.

Basi di dati e sistemi di gestione

Sistemi di gestione di basi di dati

Gli strumenti per la gestione delle basi di dati sono definiti **Data Base Management System (DBMS)**. L'acronimo indica specificatamente lo strumento con cui si gestiscono gli archivi informatici, non gli archivi stessi..

Il termine «database» è tuttavia nell'uso comune adoperato indifferentemente per riferirsi all'insieme dei dati come anche alle tecnologie che ne consentono la gestione.

Più correttamente in ambito scientifico - disciplinare è però da preferire il termine DBMS per indicare le soluzioni hardware/software e il termine base di dati per riferirsi all'insieme delle informazioni gestite dal sistema.

Perché utilizzare una base di dati e un DBMS

In linea generale possiamo affermare che

quasi tutte le operazioni informatiche
si basano sull'utilizzo di una base di dati

Se infatti ci soffermiamo sulle motivazioni che portano all'uso di un database possiamo comprendere come ***le informazioni prive di una qualsiasi minima struttura*** *siano di fatto* ***molto poco utilizzabili*** *all'interno di procedure automatizzate*.

Perchè utilizzare una base di dati e un DBMS

Sempre in linea generale una base di dati è un insieme di informazioni organizzato mediante una struttura logica che ne permette innanzi tutto la **registrazione persistente** (*tecniche di storage*), successivamente e recursivamente **l'individuazione rapida di ogni singolo elemento** al suo interno (*tecniche di query*).

Nello specifico una base di dati gestita tramite un DBMS consente inoltre un'ottimizzazione delle informazioni finalizzata all'eliminazione delle ridondanze e conseguentemente ad un utilizzo efficiente dello spazio, un mantenimento delle performance, una sufficiente affidabilità e coerenza dei contenuti.

Inoltre, in relazione alle specifiche applicazioni e agli specifici strumenti adottati, la base di dati può dover permettere l'accesso differenziato e simultaneo da parte dei diversi utenti e/o applicazioni sia in contesti personali o ristretti sia in contesti ad architettura distribuita in rete.

Impieghi tipici delle basi di dati

Come detto sopra quasi tutte le operazioni informatiche sono attualmente supportate da una base di dati. Al fine di rendere più chiaro lo scenario di utilizzo di questa tecnologia possiamo sommariamente individuare quattro classi di applicazioni che utilizzano DBMS:

- *Applicazioni "personali" o di tipo desktop*
- *Applicazioni con architettura client-server in rete locale*
- *Applicazioni con architettura client-server orientata al web (web applications)*
- *Applicazioni basate su servizi di rete (web services / architetture SOA - Service Oriented Architecture)*

Occorre tuttavia evidenziare che lo stato dell'arte permette anche un elevato grado di ibridazione tra applicazioni tipologicamente diverse secondo le tecniche di integrazione di dati e di interoperabilità. Esempi tipici di applicazioni desktop con database sono tutti i cataloghi come rubriche, bibliografie e altri archivi dove un'applicazione in esecuzione sul proprio computer si interfaccia ad un file o un insieme di files che contengono i dati organizzati secondo i criteri di ricerca previsti nell'applicazione.

Impieghi tipici delle basi di dati

Sono invece degli **esempi tipici di soluzioni ad architettura client-server** i pacchetti software dove un componente leggero per l'accesso ai dati (client) viene installato su uno o più computer e successivamente viene connesso ad un altro applicativo che si occupa di svolgere fisicamente le operazioni sui dati (server).
I software di questo tipo, solitamente orientati a contesti aziendali sono adatti ad architetture hardware basate su rete locale LAN con un numero n di postazioni di lavoro e uno o alcuni calcolatori ad alte prestazioni con funzioni di file server, database server e altre funzionalità specifiche per la rete. Tipicamente i gestionali aziendali centralizzati sono realizzati con questa architettura.

Quando l'applicazione/computer che opera sui dati alimenta uno o più siti web si parla di web application.
In questo caso, i client non sono in numero limitato ma virtualmente illimitato in quanto tutti gli utenti presenti sul web possono accedere al segmento client dell'applicazione che è sviluppata per essere utilizzata da un browser senza la necessità di installare altre componenti.

Impieghi tipici delle basi di dati

Un ulteriore **evoluzione delle applicazioni orientate alla rete internet** è data dai **servizi web** (web services).

Queste soluzioni si basano sullo spezzettamento di un'applicazione complessa in più parti funzionali semplici detti servizi.

Ogni servizio è accessibile via web tramite un **protocollo** che definisce principalmente:

a) *in che modo chiedere cosa fa il servizio;*
b) *come il servizio deve dichiarare il tipo di lavoro che fa;*
c) *come inoltrare le richieste per ottenere i risultati delle elaborazioni prodotte dal servizio.*

Una delle prerogative principali delle architetture orientate ai servizi è che il dialogo avviene sempre sul **protocollo di comunicazione internet (HTTP)** con informazioni espresse in linguaggio **XML**.

Impieghi tipici delle basi di dati

Moltissimi servizi web forniscono elaborazioni di dati gestiti da DBMS.

Uno degli esempi più efficaci è probabilmente il servizio di geocodifica di indirizzi e toponimi di Google Maps: se si digita questa URL:

http://www.google.com/maps?q=roma

si invia una richiesta ad un servizio di Google che ricerca le coordinate del toponimo «Roma» (geocoding) e le restituisce come pagina web con una mappa interattiva posizionata sulla città.

Questo è un tipico **servizio di mapping** con una definizione di protocollo molto semplice:

1. URL di base: www.google.it/maps
2. Parametro di ricerca q che accetta una stringa che «dovrebbe» corrispondere ad un toponimo.

Basi di Dati
richiami di teoria

Cos'è una Base di Dati

Una base di dati è l'insieme di dati relativo ad un sistema informativo

Cosa caratterizza una base di dati?

- **la struttura dei dati**
- **le relazioni fra i dati**

Quali sono i requisiti di una base di dati?

- **la ridondanza minima**
  - *i dati non devono essere inutilmente duplicati per problemi di spazio, gestione, manutenzione, affidabilità e coerenza*
- **la permanenza dei dati**
  - *la base di dati è protetta contro eventi che possano minacciarne l'esistenza e/o l'integrità*
- **la condivisione dei dati**
  - *più utenti devono potere ad un tempo usare la stessa base di dati (supporto unico, aggiornamento unico, coerenza dei dati, affidabilità, ...)*

Prof. M. Cossentino – Univ. di Palermo

Concetti di base

Schema di una base di dati
Descrizione della struttura dei dati di uno specifico contesto applicativo

Istanza (o occorrenza) di una base di dati
Valore assunto dai dati di un certo DB in un certo istante

Organizzazione degli archivi
Le informazioni contenute in memoria vengono organizzate in record logici

Num Record	Nome	Indirizzo	Telefono
1	Rossi Carlo	Via dei Tigli, 32	02-33249187

Un record è composto da campi
Più record possono essere contenuti in un file (o più file)

Prof. M. Cossentino – Univ. di Palermo

Concetti di base

Problemi legati alle informazioni archiviate in modo de-strutturato

1. Inconsistenza e ridondanza dei dati

Vi possono essere differenze tra i valori relativi ad una stessa entità ma registrati in posizioni diverse.
La duplicazione di dati crea ridondanze, ambiguità e difficoltà di gestione

2. Integrità dei dati

L'integrità dei dati viene assicurata dai «vincoli di consistenza»
(Ad esempio un campo non può assumere valore negativo)
Se l'informazione è registrata in più punti del sistema è necessario predisporre meccanismi talvolta complessi che ne assicurino la congruità.

3. Concorrenza

Difficoltà o impossibilità di gestire accessi simultanei alla stessa informazione da parte di più utenti o applicazioni.

Prof. M. Cossentino – Univ. di Palermo

Concetti di base

Vantaggi dei Data Base Management Systems (DBMS)

I tutti i sistemi:

1. I dati non sono duplicati e non sussistono ridondanze
2. I vincoli di consistenza fanno parte integrante della base dati e non sono applicazioni esterne

Nei sistemi complessi (es. database server):

3. L'accesso ai dati può avvenire in modo differenziato per diversi utenti secondo un sistema di privilegi fissati all'interno della base dati stessa
4. L'accesso ai dati può essere effettuato da diversi utenti simultaneamente con un sistema di controllo e mutua esclusione (blocco dei records) gestito dal DBMS

Nelle architetture:

5. I dati possono alimentare applicazioni diverse anche residenti su diverse piattaforme hardware / software

Concetti di base

Tra i vari modelli di basi dati il modello di gran lunga più utilizzato è il modello relazionale

Una base di dati relazionale è una collezione di relazioni (o tabelle)

Una relazione è una tabella costituita da un numero fisso di colonne dette attributi (o campi) e un numero variabile di righe dette “tuple” (o records)

Concetti di base

Chiave in una relazione

E' un sottoinsieme K degli attributi che soddisfa le proprietà di:

unicità: in qualunque istanza della relazione, non possono esistere due tuple distinte la cui restrizione su K sia uguale

minimalità: non è possibile sottrarre alla relazione un attributo senza violare la condizione di unicità

In generale una relazione può avere più di una chiave

Ad esempio una relazione contenente nomi di persone e caratterizzata dai seguenti attributi: [codice fiscale, nome, cognome, codice meccanografico anagrafe, data di nascita] possiede due chiavi.

Chiave primaria in una relazione

Corrisponde alla chiave usata più frequentemente per accedere ai dati

Progettazione di una base di dati

Le fasi di progettazione di un DB sono tre:

1 - Progetto CONCETTUALE (-> Schema concettuale)
2 - Progetto LOGICO (-> Schema logico)
3 - Progetto FISICO (-> Schema fisico)

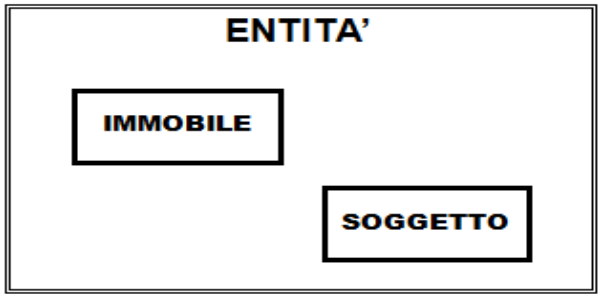
1 - Progetto CONCETTUALE
Lo schema concettuale è la rappresentazione più astratta, la più vicina alla logica umana nella definizione di dati e relazioni.
Spesso vengono usati i modelli entità-relazioni (ERD)

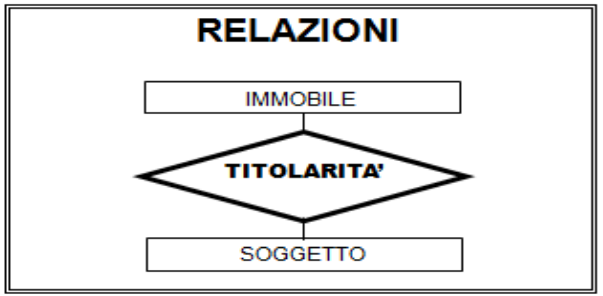
2 - Progetto LOGICO
Lo schema logico dipende fortemente dal DBMS e dal suo modello logico dei dati. Esistono ad esempio DBMS gerarchici, reticolari e relazionali. Nello schema logico vengono definite anche le viste (dette anche schemi esterni) cioè le parti del DB messe a disposizione delle applicazioni.

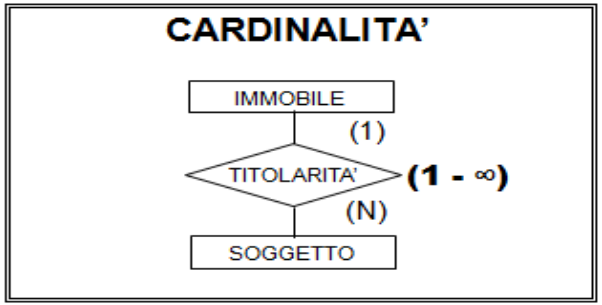
3 - Progetto FISICO
Lo schema fisico definisce come le strutture definite nel progetto logico vanno implementate nell'archivio e nel file system scelti.

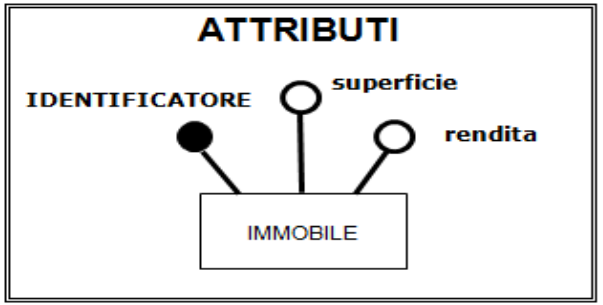
Prof. M. Cossentino – Univ. di Palermo

Modello Entity-Relationship (E-R)









Definizioni

Entità: un qualsiasi oggetto concettuale che rappresenta una data informazione e che può essere individuato e distinto dagli altri.
Attributi: insieme di valori che caratterizzano un'entità
Attributi chiave: insieme degli attributi sufficienti ad identificare univocamente un'entità all'interno di un certo insieme

Rappresentazione del modello E-R (ERD)

Dal modello ERD al modello fisico
Regole per passare dallo schema concettuale (ERD) allo schema logico relazionale (tabella):

1 - Ad ogni entità dell'ERD corrisponde una tabella

2 - Le relazioni fra entità dell'ERD (tabelle T1 e T2) sono di fatto costituite da particolari colonne in comune (chiavi primarie ed esterne)

2a - Se la relazione è **1:1** agli attributi di T1 si possono aggiungere quelli che sono chiave primaria di T2 (chiave esterna di T2) e viceversa

2b - Se la relazione è **1:N** agli attributi di T2 si possono aggiungere quelli che sono chiave primaria di T1 (chiave esterna di T2) e non viceversa

2c - Se la relazione è **N:N** è necessario creare una tabella T3 le cui colonne sono le chiavi primarie di T1 e T2. La chiave primaria di T3 è l'insieme delle sue colonne ed è l'insieme delle chiavi esterne di T1 e T2

Prof. M. Cossentino – Univ. di Palermo

Tipi di dati e informazioni

Dato ASCII numerico

sotto-tipi: intero, decimale, virgola mobile, data/ora, contatori, codifiche …

Dato ASCII alfanumerico

sotto-tipi: testo (dal singolo carattere a lunghezza non finita), codifiche …

Dato binario (BLOB)

sotto-tipi: immagini, conversione di files, qualsiasi sequenza di bit.

SQL – Structured Query Language

Interagire con una base di dati

I comandi con cui si può interagire con una base di dati all'interno di un DBMS appartengono al linguaggio

SQL

(**Structured Query Language**)

Qualsiasi DBMS è dotato di una *shell* con la quale è possibile inviare comandi più o meno complessi scritti in SQL. Il numero e il tipo di comandi disponibili dipende dal DBMS utilizzato, tuttavia il *core* dei comandi SQL è relativamente standardizzato.

Normalizzazione dei dati per l'utilizzo di DBMS

Prima forma normale

Edificio	Unità immobiliari			
Edificio 1	UI1	UI2	UI3	
Edificio 2	UI4	UI5		

In una tabella di dati ogni colonna deve assumere un solo valore, ovvero non può essere una matrice di valori

Edificio	Unità immobiliari
Edificio 1	UI1
Edificio 1	UI2
Edificio 1	UI3
Edificio 2	UI4
Edificio 2	UI5

In questo caso la normalizzazione consiste nel riportare le celle che originariamente erano raggruppate in una unica colonna in più righe replicando gli altri valori

Normalizzazione dei dati per l'utilizzo di DBMS

Seconda forma normale

in una tabella con chiave a più attributi, ogni colonna non appartenente alla chiave deve dipendere da tutte le colonne chiave e non solo da una parte di esse.

Codice città	Codice via	Città	Via
01	1	Roma	Cesare Battisti
01	2	Roma	Cavour
03	1	Venezia	Verdi

"Città" dipende soltanto da "codice città"

Codice città	Città
01	Roma
03	Venezia

Via	Codice via	Codice città
Cesare Battisti	1	01
Cavour	2	01
Verdi	1	03

In questo caso la normalizzazione consiste nel produrre nuove tabelle che soddisfino la condizione di partenza

Normalizzazione dei dati per l'utilizzo di DBMS

Terza forma normale

In una tabella la dipendenza fra le colonne deve essere basata soltanto sulla chiave primaria.

Codice zona	Tipo zona	Indice densità
01	C1	2,0
02	C1	2,0
03	C2-1	1,0
04	C2-1	1,0
05	C2-2	0,85

"Indice densità" dipende dal tipo di zona non dal codice zona

Codice zona	Tipo zona
01	C1
02	C1
03	C2-1
04	C2-1
05	C2-2

Tipo zona	Indice densità
C1	2,0
C2-1	1,0
C2-2	0,85

Anche in questo caso la normalizzazione consiste nel produrre nuove tabelle che soddisfino la condizione di partenza.

Utilizzo della base di dati: le query di selezione

Con gli operatori **select** e **from** si estraggono informazioni da una tabella.

In questo esempio il risultato della query è identico alla tabella di origine

Comuni

Comune	Cod_ provincia
Scorzè	VE
Caorle	VE
Montecchio Vic.	VI
San Martino Buon Albergo	VR

Select * from comuni <- (Selezione con SQL)

Comune	Cod_ provincia
Scorzè	VE
Caorle	VE
Montecchio Vic.	VI
San Martino Buon Albergo	VR

Utilizzo della base di dati: le query di selezione

Sostituendo l'asterisco con una sequenza di nomi di campi si estraggono alcune colonne di una tabella.

(il numero delle righe rimane inalterato)

Comuni

Comune	Cod_ provincia
Scorzè	VE
Caorle	VE
Montecchio Vic.	VI
San Martino Buon Albergo	VR

Select comune from comuni (Selezione con SQL)

Comune
Scorzè
Caorle
Montecchio Vic.
San Martino Buon Albergo

Utilizzo della base di dati: le query di selezione

Con l'uso della clausola "where" il risultato della query è costituito da righe che sono un sottoinsieme della tabella di origine (questa operazione si può anche definire un "filtro" sul contenuto della tabella.)

Comuni

Comune	Cod_ provincia
Scorzè	VE
Caorle	VE
Montecchio Vic.	VI
San Martino Buon Albergo	VR

Select * from comuni where cod_provincia = 'VE'

(Selezione con SQL)

Comune	Cod_provincia
Scorzè	VE
Caorle	VE

Utilizzo della base di dati: le query di selezione

Comuni

Comune	Cod_provincia
Scorzè	VE
Caorle	VE
Montecchio Vic.	VI
San Martino Buon Albergo	VR

Province

Cod_Provincia	Provincia
VE	Venezia
VI	Vicenza
PD	Padova

Select * from comuni, province

Comune	Cod_ provincia	Cod_ provincia	Provincia
Scorzè	VE	VE	Venezia
Caorle	VE	VE	Venezia
Montecchio Vic.	VI	VE	Venezia
San Martino Buon Albergo	VR	VE	Venezia
Scorzè	VE	VI	Vicenza
Caorle	VE	VI	Vicenza
Montecchio Vic.	VI	VI	Vicenza
San Martino Buon Albergo	VR	VI	Vicenza
Scorzè	VE	PD	Padova
Caorle	VE	PD	Padova
Montecchio Vic.	VI	PD	Padova
San Martino Buon Albergo	VR	PD	Padova

Una query con due tabelle estrae tutte le combinazioni possibili tra i rispettivi record

Record i cui i valori di codice provincia coincidono

Utilizzo della base di dati: le query di selezione

In una query con più tabelle, utilizzando la clausola where ottengo un risultato più utile escludendo le combinazioni comune-provincia che non hanno senso

Comuni

Comune	Cod_provincia
Scorzè	VE
Caorle	VE
Montecchio Vic.	VI
San Martino Buon Albergo	VR

Province

Cod_Provincia	Provincia
VE	Venezia
VI	Vicenza
PD	Padova

Select * from comuni, province
where comuni.cod_provincia = province.cod_provincia

Comune	Cod_ provincia	Cod_ provincia	Provincia
Scorzè	VE	VE	Venezia
Caorle	VE	VE	Venezia
Montecchio Vic.	VI	VI	Vicenza

Utilizzo della base di dati: le query di selezione

Utilizzando l'operatore di join (tipo inner join) ottengo un risultato analogo al precedente.

NB: In entrambi i casi i records che non hanno riferimenti non vengono considerati (cfr San Martino e Padova)

Comuni

Comune	Cod_provincia
Scorzè	VE
Caorle	VE
Montecchio Vic.	VI
San Martino Buon Albergo	VR

Province

Cod_Provincia	Provincia
VE	Venezia
VI	Vicenza
PD	Padova

```
Select * from comuni
inner join province on comuni.cod_provincia =
province.cod_provincia
```

Comune	Cod_ provincia	Cod_ provincia	Provincia
Scorzè	VE	VE	Venezia
Caorle	VE	VE	Venezia
Montecchio Vic.	VI	VI	Vicenza

Utilizzo della base di dati: le query di selezione

Utilizzando il join di tipo left ottengo un risultato equivalente all'inner join a cui però vengono aggiunti i record della tabella di sinistra che non hanno riferimenti; su queste righe i campi provenienti alla tabella di destra non riportano valori ma il cosiddetto "null" (nessun valore).

Comuni

Comune	Cod_provincia
Scorzè	VE
Caorle	VE
Montecchio Vic.	VI
San Martino Buon Albergo	VR

Province

Cod_Provincia	Provincia
VE	Venezia
VI	Vicenza
PD	Padova

```
Select * from comuni
left join province on comuni.cod_provincia =
province.cod_provincia
```

Comune	Cod_ provincia	Cod_ provincia	Provincia
Scorzè	VE	VE	Venezia
Caorle	VE	VE	Venezia
Montecchio Vic.	VI	VI	Vicenza
San Martino Buon Albergo	VR		

Utilizzo della base di dati: le query di selezione

Utilizzando il join di tipo right ottengo un risultato equivalente all'inner join a cui però vengono aggiunti i record della tabella di destra che non hanno riferimenti.

Comuni

Comune	Cod_ provincia
Scorzè	VE
Caorle	VE
Montecchio Vic.	VI
San Martino Buon Albergo	VR

Province

Cod._Provincia	Provincia
VE	Venezia
VI	Vicenza
PD	Padova

```
Select * from comuni
right join province on comuni.cod_provincia =
province.cod_provincia
```

Comune	Cod_ provincia	Cod_ provincia	Provincia
Scorzè	VE	VE	Venezia
Caorle	VE	VE	Venezia
Montecchio Vic.	VI	VI	Vicenza
		PD	Padova

Utilizzo della base di dati: gli ordinamenti

```
Select * from comuni
order by comune
```

Ordinamento decrescente:

```
Select * from comuni
order by comune desc
```

Ordinamento multiplo:

```
Select * from comuni
order by cod_provincia, comune
```

Utilizzo della base di dati: gli operatori di aggregazione

```
Select cod_provincia from comuni
group by cod_provincia
```

Conteggio:

```
Select count(cod_provincia) from comuni
group by cod_provincia
```

Operatori statistici: somma, media, minimo, massimo:

Sum(), Avg(), Min(), Max()

Filtro sul risultato aggregato:

```
Select cod_provincia from comuni
group by provincia
having cod_provincia <> 'VE'
```

Utilizzo della base di dati: le query di comando

Creazione tabella

Select * **into** comuni2 from comuni

Aggiornamento

Update comuni **set** abitanti = 10000

Accodamento

Insert into comuni (comune, cod_provincia) **values** (‘Scorzè’, ‘VE’)

Eliminazione

Delete from comuni where cod_provincia=‘TN'

Altri operatori come “ALTER TABLE”, “CREATE TABLE”, e altri ancora sono disponibili nella maggioranza dei DBMS ma non in tutti (ad es. in MS Access non è disponibile come comando SQL ma solo come funzione di interfaccia).