

Metodi per la Gestione dei Dati

(prof. Mauro Leoncini e Massimo Santini)

Anno Accademico 2002/03

Riassunto a cura di: Fabio Ruini

(ruini.fabio@unimore.it)

Parte I.1 – “Concetti introduttivi sulle basi di dati”

✚ *Sistema informativo*: proprio di ogni organizzazione, anche se può non essere esplicitato nella struttura, svolge le seguenti funzioni:

1. raccolta ed acquisizione delle informazioni;
2. archiviazione e conservazione delle informazioni;
3. elaborazione delle informazioni;
4. distribuzione e scambio delle informazioni.

L'esistenza di un sistema informativo è indipendente dalla sua automatizzazione.

✚ *Sistema informatico*: porzione automatizzata del sistema informativo; si tratta di quel sottosistema che gestisce informazioni per mezzo delle tecnologie informatiche.

Nei sistemi informatici, le informazioni vengono rappresentate per mezzo di *dati*, i quali necessitano di una interpretazione per fornire *informazioni*. Sebbene sia difficile fornire definizioni precise per i due termini succitati, lo Zingarelli fornisce la seguente classificazione:

✚ *Dato*: ciò che è immediatamente presente alla conoscenza, prima di ogni elaborazione; (in informatica) elementi di informazione costituiti da simboli che debbono essere elaborati;

✚ *Informazione*: notizia, dato o elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni, modi d'essere.

Dal concetto di “dato”, deriva direttamente quello di “base di dati”, che è possibile indagare secondo due differenti punti di vista:

✚ *Base di dati – accezione generica*: collezione di dati, utilizzati per rappresentare le informazioni di interesse per una o più applicazioni di una organizzazione;

✚ *Base di dati – accezione tecnica*: collezione di dati gestita da un “Data Base Management System” (DBMS)

L'ultimo concetto introdotto, quello di DBMS, è definibile come:

✚ *DBMS*: sistema software in grado di gestire collezioni di dati che siano di elevate dimensioni (molto maggiori rispetto alla memoria centrale disponibile, con conseguente necessario ricorso alla memoria secondaria), condivise (utilizzabili contemporaneamente da programmi diversi) e persistenti (cioè con un tempo di vita che non sia limitato a quello delle singole

esecuzioni dei programmi che le utilizzano), assicurando la loro affidabilità (capacità di conservare sostanzialmente intatto il contenuto della base di dati in caso di malfunzionamenti hardware e software) e riservatezza (ciascun utente è abilitato a svolgere solo determinate azioni sui dati, mediante meccanismi di autorizzazione). Come ogni prodotto informatico, un DBMS deve essere efficiente (capace di svolgere le operazioni utilizzando un insieme di risorse spazio/temporali accettabili per l'utente) ed efficace (capace di rendere produttive le attività dei suoi utenti).

I DBMS gestiscono insiemi di dati strutturati (fisicamente memorizzati in files di registrazioni, altresì detti record, di formato omogeneo) e non possono pertanto operare con basi di dati non strutturati (collezione di dati memorizzati come documenti anziché come record), sulle quali agiscono invece gli IRS (Information Retrieval Systems).

Affinché i dati di interesse siano correttamente organizzati e la loro struttura sia descritta in un modo comprensibile per l'elaboratore, si ricorre ad un insieme di concetti detto "modello dei dati". Esistono due categorie principali di tali modelli:

- + *Modelli logici*: utilizzati nei DBMS esistenti per l'organizzazione dei dati, ad essi fanno riferimento i programmi. I modelli logici utilizzano strutture che, pur astratte, riflettono una particolare organizzazione logica dei dati (ad alberi, a grafi, a tabelle o ad oggetti) ed è per questo motivo che la conoscenza del modello logico è necessaria per l'utilizzo di una base di dati;
- + *Modelli concettuali*: permettono di rappresentare i dati in maniera completamente indipendente da ogni sistema (ad es. dal modello logico). Il termine "concettuale" deriva dal fatto che tali modelli tendono a descrivere concetti del mondo reale, piuttosto che i dati utili per rappresentarli.

Al giorno d'oggi, il modello logico più diffuso è quello relazionale. In esso, i dati sono definiti mediante "relazioni" ed organizzati in insiemi di record aventi struttura fissa. Le relazioni vengono spesso rappresentate graficamente mediante tabelle, dove ogni riga corrisponde ad un record ed ogni colonna rappresenta un campo del record. Una base di dati relazionale comprende in genere più relazioni.

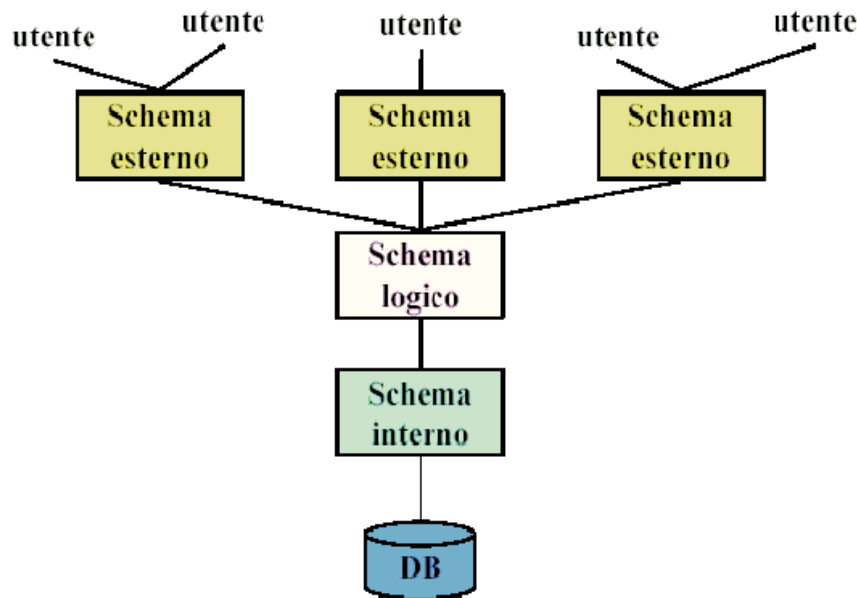
Nelle basi di dati è possibile distinguere sostanzialmente tra due componenti:

- + *Schema della BD*: corrispondente alla parte sostanzialmente invariante nel tempo, costituita dalle caratteristiche dei dati;
- + *Istanza della BD*: corrispondente alla parte variabile nel tempo, costituita dai valori effettivi.

Esiste inoltre una proposta di architettura standardizzata per DBMS, articolata su tre livelli, detti rispettivamente esterno, logico e fisico, a ciascuno dei quali corrisponde un particolare schema:

- + *Schema logico*: costituisce una descrizione dell'intera base di dati per mezzo del modello logico adottato dal DBMS (cioè tramite uno dei seguenti modelli: relazionale, gerarchico, reticolare o ad oggetti);
- + *Schema fisico*: costituisce la rappresentazione dello schema logico per mezzo di strutture fisiche di memorizzazione. Ad esempio, una relazione può essere realizzata fisicamente per mezzo di un file sequenziale, o di un file hash, o di un file sequenziale con uno o più indici;

- ✚ *Schema esterno*: costituisce la descrizione di una porzione della base di dati di interesse, per mezzo del modello logico. Uno schema esterno può prevedere organizzazioni dei dati diverse rispetto a quelle utilizzate nello schema logico, che riflettono il punto di vista di un particolare utente o insieme di utenti. E' pertanto possibile associare ad uno schema logico vari schemi esterni, come mostra la figura qui sotto:



La conseguenza più importante dell'articolazione in livelli è la cosiddetta indipendenza dei dati. Un'indipendenza che può essere di due tipi:

- ✚ *Indipendenza fisica*: il livello logico (e, di conseguenza, anche quello esterno) è indipendente dal livello fisico. Ciò consente di interagire con il DBMS in modo indipendente dalla struttura fisica dei dati (ad esempio modificando l'allocazione fisica dei file sui dispositivi di memorizzazione, senza influire sulle descrizioni dei dati ad alto livello e quindi sui programmi che utilizzano gli stessi);
- ✚ *Indipendenza logica*: il livello logico è indipendente dal livello esterno, ed è pertanto possibile interagire con quest'ultimo (ad esempio, aggiungendo uno schema esterno in base alle esigenze di un nuovo utente), senza dover modificare lo schema logico (e, di riflesso, senza intaccare l'organizzazione fisica dei dati).

Su un DBMS è possibile specificare operazioni di vario tipo, in particolare relative agli schemi ed alle istanze. I linguaggi per basi di dati si dividono in tre categorie:

- ✚ *Data Definition Language (DDL)*: per la definizione di schemi (esterni, logici e fisici) ed altre operazioni generali;
- ✚ *Data Manipulation Language (DML)*: per l'interrogazione e l'aggiornamento delle istanze di basi di dati;
- ✚ *Query Language (QL)*: esclusivamente per l'interrogazione.

Tra gli utenti di DBMS, assume particolare importanza la categoria degli “utenti finali” (altresì detti “terminalisti”), che eseguono operazioni predefinite, chiamate transazioni:

✚ *Transazione*: programma che realizza attività frequenti e predefinite, con poche eccezioni, comunque previste a priori.

L'utilizzo di un DBMS all'interno di un'organizzazione, porta con sé diversi vantaggi, ma è accompagnato allo stesso tempo da alcuni aspetti negativi. I vantaggi principali possono ricondursi principalmente all'indipendenza dei dati (che favorisce lo sviluppo di applicazioni più flessibili e facilmente modificabili) ed alla loro condivisione (che riduce così il rischio di ridondanza e di incoerenza dei dati stessi). Viceversa, gli svantaggi sono da ricercare fundamentalmente tra i costi, che possono essere molto elevati (sia per quanto riguarda quelli diretti, cioè il DBMS, sia per quelli indiretti, ovvero macchine, formazione del personale, ecc...).

Parte I.2 – “Modello relazionale dei dati”

Il modello relazionale si basa su due concetti, relazione e tabella, di natura diversa, ma facilmente riconducibili l'uno all'altro. Il secondo è intuitivo ed identificabile come semplice struttura composta da righe e colonne. Il termine relazione, può invece essere visto sotto tre diversi punti di vista:

✚ *Relazione matematica*, come deriva dalla teoria degli insiemi;

✚ *Relazione come traduzione di relationship*, costruito del modello concettuale Entity-Relationship, utilizzato per descrivere una correlazione o associazione tra due entità;

✚ *Relazione, secondo la definizione del modello relazionale dei dati*.

Per analizzare l'accezione matematica del termine “relazione”, derivante dalla teoria degli insiemi, occorre richiamare alcune nozioni elementari:

✚ *Prodotto cartesiano*: dati due insiemi D_1 e D_2 , il prodotto cartesiano (in simboli, $D_1 \times D_2$) è l'insieme delle coppie ordinate (v_1, v_2) , tali che v_1 è un elemento di D_1 e v_2 è un elemento di D_2 ;

Ne deriva che una relazione matematica sugli insiemi D_1, D_2, \dots, D_n (chiamati “domini” della relazione) è un qualunque sottoinsieme del prodotto cartesiano $D_1 \times D_2 \times \dots \times D_n$. Le relazioni possono essere rappresentate graficamente, in maniera utilmente espressiva, sotto forma tabellare. Per ogni relazione matematica è possibile identificare due proprietà:

✚ *Grado della relazione*: numero n delle componenti del prodotto cartesiano (e quindi di ogni n-upla);

✚ *Cardinalità della relazione*: numero di elementi (cioè di n-uple) della relazione.

Nell'ambito delle relazioni matematiche, ciascuna n-upla è, al proprio interno, ordinata: l' i -esimo valore di ciascuna proviene dell' i -esimo dominio (struttura posizionale). L'ordinamento appena evidenziato tra i domini di una relazione, corrisponde in effetti ad una caratteristica insoddisfacente del concetto di relazione matematica rispetto alla

possibilità di organizzare ed utilizzare i dati. In vari contesti dell'informatica si tende infatti a privilegiare notazioni non posizionali, rispetto a quelle posizionali. Il problema è risolvibile associando a ciascuna occorrenza di dominio nella relazione un nome, detto attributo, che descrive il "ruolo" giocato dal dominio stesso. Nella rappresentazione tabellare, possiamo utilizzare gli attributi come intestazioni per le colonne. L'ordinamento degli attributi diventa così irrilevante: non è più necessario parlare di primo dominio, secondo dominio e così via, ma è sufficiente far riferimento agli attributi. Nel modello relazionale dei dati, gli elementi della relazione vengono detti "tuple", così definite:

- ✚ *Tupla*: funzione che associa ad ogni attributo un elemento del dominio corrispondente. Una tupla su un insieme di attributi X è una funzione che associa, a ciascun attributo A in X, un valore del dominio d(A).

Secondo il modello relazionale dei dati, una relazione su X è dunque un insieme di tuple su X. Normalmente, una base di dati relazionale è costituita da più relazioni, le cui tuple contengono valori comuni. Una delle caratteristiche fondamentali del modello relazionale viene spesso indicata dicendo che esso è appunto "basato su valori": i riferimenti fra dati in relazioni diverse sono rappresentati per mezzo di valori dei domini che compaiono nelle tuple. Un esempio di base di dati relazionale è il seguente:

studenti

Matricola	Cognome	Nome	Data di Nascita
23146	Rossi	Paolo	5/12/1978
23148	Bianchi	Anna	3/11/1976
23149	Verdi	Aldo	12/11/1979
23152	Neri	Carla	1/2/1978

esami

Studente	Voto	Corso
23152	30	04
23152	24	02
23149	28	01
23146	26	01

corsi

Codice	Titolo	Docente
01	Economia politica	Neri
02	Diritto privato	Bruni
04	Informatica	Verdi

Possiamo a questo punto formalizzare le definizioni relative al modello relazionale, distinguendo il livello degli schemi da quello delle istanze:

- ✚ *Schema di relazione*: costituito da un simbolo R, detto "nome della relazione", e da un insieme di (nomi di) attributi $X = \{ A_1, A_2, \dots, A_n \}$, il tutto di solito indicato con $R(X)$;
- ✚ *Schema di base di dati*: insieme di schemi di relazione con nomi diversi, indicato di solito con $R = \{ R_1(X_1), R_2(X_2), \dots, R_n(X_n) \}$;
- ✚ (istanza di) *Relazione su uno schema $R(X)$* : insieme r di tuple su X;

- ✚ (istanza di) Base di dati su uno schema $R = \{ R_1(X_1), R_2(X_2), \dots, R_n(X_n) \}$: insieme di relazioni $r = \{ r_1, r_2, \dots, r_n \}$, dove ogni r_i , per $1 \leq i \leq n$, è una relazione sullo schema $R_i(X_i)$.

La struttura del modello relazione è indubbiamente molto semplice e potente, ma al tempo stesso impone un certo grado di rigidità, in quanto le informazioni debbono essere rappresentate per mezzo di tuple di dati omogenee. In molti casi reali, tuttavia, i dati disponibili possono non corrispondere esattamente al formato previsto (per esempio, per la mancanza dell'attributo "NumeroDiTelefono" in una relazione "Persone"). Per ovviare a questo inconveniente, il concetto di relazione viene di solito esteso prevedendo che una tupla possa assumere, su ciascun attributo, un valore speciale (solitamente non appartenente al dominio) detto "valore nullo", che denota l'assenza di informazione. Relativamente ai valori nulli, possono essere determinate sostanzialmente tre circostanze:

- ✚ *Valore sconosciuto*: ad esempio, non si conosce il numero di telefono di una persona;
- ✚ *Valore inesistente*: ad esempio, una studentessa non ha obblighi militari ed è pertanto inesistente il valore del dominio corrispondente);
- ✚ *Valore senza informazione*: data dalla disgiunzione logica ("or") delle due situazioni precedenti.

Nei sistemi di basi di dati relazioni è di solito prevista una gestione molto semplice, ma tutto sommato efficace, del valore nullo, sul quale non viene fatta alcuna ipotesi e quindi, in pratica, ci si trova nell'ultimo caso, ossia quello del valore senza informazione. In molti casi, tuttavia, non è vero che qualsiasi insieme di tuple sullo schema rappresenti informazioni corrette per l'applicazione. E' stato pertanto introdotto il concetto di "vincolo di integrità", così definito:

- ✚ *Vincolo di integrità*: proprietà che deve essere soddisfatta dalle istanze che rappresentano informazioni corrette per l'applicazione.

Ogni vincolo può essere visto come un predicato che associa ad ogni istanza il valore "vero" o "falso" Se il predicato assume il valore vero, diciamo che l'istanza soddisfa il vincolo. In generale, ad uno schema di base di dati associamo un insieme di vincoli e consideriamo corrette le istanze che soddisfano tutti questi vincoli. I vincoli di integrità possono essere classificati in questo modo:

- ✚ *Vincoli intrarelazionali*: se il loro soddisfacimento è definito rispetto a singole relazione della base di dati. Possiamo ulteriormente suddividere i vincoli intrarelazionali:
 - ✚ *Vincoli di tupla*: esprimono condizioni sui valori di ciascuna tupla, indipendentemente dalle altre. Ad esempio, su una relazione sullo schema: PAGAMENTI (Importo, Ritenute, Netto), un vincolo di tupla potrebbe essere quello che impone $\text{Netto} = \text{Importo} - \text{Ritenute}$;
 - ✚ *Vincoli di dominio*: caso particolare di vincolo di tupla che coinvolge un solo attributo. Ad esempio: $(\text{Voto} \geq 18) \text{ AND } (\text{Voto} \leq 30)$;
 - ✚ *Vincoli di chiave*: si tratta dei vincoli più importanti del modello relazionale, senza i quali il modello stesso non avrebbe senso. Essi si basano sul concetto di "chiave", che intuitivamente è un insieme di attributi utilizzato per identificare

univocamente le tuple di una relazione. Formalizzando la definizione, procediamo in due passi:

- *Superchiave*: un insieme K di attributi è superchiave di una relazione r , se r non contiene due tuple distinte t_1 e t_2 , con $t_1[K] = t_2[K]$;
- *Chiave*: K è chiave di r se è una superchiave minimale di r , cioè non esiste un'altra superchiave K' di r che sia contenuta in K come sottoinsieme proprio.

Ciascuna relazione e ciascuno schema di relazione hanno sempre una chiave. Questa caratteristica deriva dal fatto che una relazione è un insieme (e, come tale, formato da elementi diversi) di tuple ed è pertanto sempre possibile identificare una superchiave formata da tutti gli attributi. Se questa superchiave è minimale, essa diventa ovviamente una chiave. Rifacendosi al discorso precedente sull'assenza di informazione, è chiaro che in presenza di valori nulli non è più vero che i valori delle chiavi permettono di identificare univocamente le tuple delle relazioni e di stabilire riferimenti fra tuple di relazioni diverse. Vi è quindi la necessità di limitare la presenza di valori nulli nelle chiavi delle relazioni; la soluzione è semplice: su una delle chiavi (detta chiave primaria) si vieta esplicitamente la presenza di valori nulli.

✚ *Vincoli interrelazionali*: se coinvolgono più relazioni. La più importante classe di vincoli interrelazioni è quella dei vincoli di integrità referenziale (foreign keys), così definiti:

- *Vincolo di integrità referenziale*: fra un insieme di attributi X di una relazione R_1 ed un'altra relazione R_2 , è soddisfatto se i valori su X di ciascuna tupla dell'istanza R_1 compaiono come valori della chiave (primaria) dell'istanza di R_2 .

Per meglio comprendere questa definizione è opportuno sottolineare come i dati presenti in relazioni diverse siano correlati attraverso valori comuni, che sono solitamente i valori delle chiavi primarie. I vincoli di integrità referenziale, in conclusione, giocano un ruolo fondamentale nel concetto di modello relazionale basato su valori.

Parte I.3 – “Linguaggio SQL”

SQL, che rappresentava originariamente l'acronimo di Structured Query Language, è il linguaggio di riferimento per le basi di dati relazionali. Il modello relazionale consente di definire formalmente le operazioni base sulle relazioni e la loro semantica. E' comunque bene chiarire cosa si intenda con il concetto di “operazione”, che è così definito:

✚ *Operazione*: funzione che, applicata ad una o più relazioni (dati), fornisce come risultato una relazione.

Le operazioni consentite dal linguaggio SQL sono numerose:

✚ *Proiezione*:

```
select lista_di_attributi
```

from tabella;

✚ *Selezione:*

```
select lista_di_attributi
from tabella
where condizione_logica;
```

✚ *Natural Join:*

```
select *
from tabella1
natural join tabella2;
```

Il join naturale dà origine ad una relazione, in cui gli attributi sono l'insieme degli attributi delle due relazioni di origine (compresi eventuali doppi). Concettualmente, il natural join consiste nel calcolare dapprima il prodotto cartesiano delle relazioni, quindi eliminare le tuple non significative (quelle, cioè, in cui i valori corrispondenti ad attributi con lo stesso nome sono diversi).

✚ *Theta join:*

```
select lista_di_attributi
from tabella1
inner join tabella2
on condizione_di_join;
```

(sintassi alternativa)
select lista_di_attributi
from tabella1, tabella2
where condizione_di_join

A differenza del join naturale, che si basa sull'uguaglianza di nomi (attributi), nel theta join è possibile specificare esplicitamente le condizioni per poter fondere le righe.

✚ *Ridenominazione:*

```
select attributo as nuovo_nome
from tabella;
```

✚ *Qualificazione:*

```
select tabella2.attributo as nuovo_nome
from tabella1;
```

✚ *Operatori insiemistici:*

```
select lista_di_attributi_1
from tabella1
where condizione1
union | intersect | except
```

```
select lista_di_attributi_1
from tabella2
where condizione2;
```

Gli operatori “union”, “intersect” ed “except” corrispondono rispettivamente all’unione, all’intersezione ed alla differenza insiemistiche. Attualmente, MySQL non riconosce nessuno di questi operatori, ma è comunque possibile ovviare a tale mancanza, con un utilizzo accorto dell’operatore select, arricchito dal parametro “distinct”.

Distinct:

```
select distinct nome_attributo
from tabella
where condizione_logica;
```

Il parametro “distinct” permette di eliminare quelle tuple che hanno identico valore sull’attributo specificato nella target list dell’istruzione select.

Operatori aggregati:

```
select
count(*) | count(distinct nome_attributo) | max(nome_attributo) | avg(nome_attributo)
from tabella
where condizione_logica;
```

I quattro operatori aggregati presenti nella sintassi di esempio di cui sopra, svolgono rispettivamente le seguenti funzioni: conteggio del numero di righe della tabella, conteggio delle righe diverse sull’attributo specificato, restituzione del valore più grande per l’attributo specificato, restituzione del valore medio per l’attributo specificato.

Raggruppamento:

```
select lista_di_attributi
from tabella
group by nome_attributo;
```

La clausola “group by” è molto utile in abbinamento agli operatori aggregati, in quanto questi ultimi operano sull’intera tabella. Con l’aggiunta della clausola di raggruppamento è possibile ampliare le possibilità offerte dagli operatori aggregati, facendoli agire distintamente su sottoinsiemi di righe.

Query annidate:

```
select attributo
from tabella
where attributo < | > | = | >= | <= | <>
any | all (select attributo from tabella);
```

Le query annidate servono a risolvere quelle situazioni in cui la condizione logica di una query (clausola “where”) non dipende da semplici valori degli attributi, ma bensì da insiemi di valori. Gli operatori any ed all rendono l’espressione complessiva vera,

rispettivamente nel caso in cui una qualunque delle espressioni sia vera (operatore any) o nell'ipotesi per cui tutte le espressioni sono vere (operatore all).

Join esterno:

```
select lista_di_attributi
from tabella1 LEFT | RIGHT | FULL join tabella2
on condizione;
```

Il join esterno fa sì che, in un'operazione di join, tutte le tuple di una relazione contribuiscano al risultato, eventualmente estese con valori nulli. Vi sono tre tipi di join esterno: sinistro (estende le tuple del primo operando), destro (estende le tuple del secondo operando) e completo (estende le tuple di entrambi gli operandi).

Ricerca all'interno di attributi di testo:

```
select attributo
from tabella
where attributo like 'sottostringa';
```

L'operatore like consente di "filtrare" i risultati di una query effettuata su campi di testo. A tal scopo, nella variabile "sottostringa" vengono utilizzati due particolari "caratteri jolly", che sono "%" (sostituisce una qualsiasi sequenza di caratteri) e "_" (che sostituisce un singolo carattere).

Ordinamento di tabelle:

```
select lista_di_attributi
from tabella
order by attributo1, attributo2, ... ASC | DESC;
```

La clausola "order by" permette di ordinare le righe di una tabella, in base ad un determinato attributo. L'ordinamento può essere crescente (funzione di default) o decrescente ed è inoltre possibile specificare più attributi sui quali effettuare l'operazione (si ordina per il primo attributo e, nel caso per tale attributo vi fossero valori uguali, si ordina in base al secondo e così via in maniera ricorsiva).

Indici:

```
create index nome_indice
on nome_tabella(nome_attributo);
```

Un indice I per una tabella T è essenzialmente una tabella ordinata rispetto ad uno o più attributi di T. Esso contiene due soli attributi e la sua funzione è sostanzialmente quella di facilitare l'accesso alle righe di T.

Definizione dei dati:

```
create table nome_tabella(
    nome_attributo dominio [default] [vincoli]
)
```

L'istruzione "create table" definisce uno schema di relazione e ne crea un'istanza vuota. E' possibile definire inoltre gli attributi ed i relativi domini, gli eventuali valori di default, gli eventuali vincoli per gli attributi e gli altrettanto eventuali vincoli tupla ed interrelazionali.

✚ *Operazioni di aggiornamento:*

```
insert into nome_tabella values lista_di_valori;  
delete from nome_tabella where condizione;  
update nome_tabella set nome_attributo = espressione where condizione;
```

Le tre operazioni succitate permettono rispettivamente di inserire nuove tuple in una relazione, di cancellare tuple esistenti o eventualmente aggiornarle/modificarle.

Parte I.4 – "Progettazione di DB relazionali"

La progettazione di basi di dati è una delle attività del processo di sviluppo dei sistemi informativi e va quindi inquadrata in un contesto più generale, che è quello del ciclo di vita dei sistemi informativi (caratterizzato da un insieme di attività iterative e quindi cicliche). L'attività di progettazione può essere scomposta tre fasi, ciascuna delle quali produce uno schema: progettazione concettuale (l'output è lo schema concettuale), progettazione logica (schema logico) e progettazione fisica (schema fisico). Tali schemi vengono costruiti in accordo a ben precisi "modelli"; in particolare occorre sottolineare l'importanza dei:

✚ *Modelli concettuali:* permettono di rappresentare i dati in modo indipendente da ogni sistema. Essi cercano di descrivere i concetti del mondo reale e sono pertanto utilizzati nelle fasi preliminari di progettazione. Il modello concettuale più utilizzato è il cosiddetto modello Entity-Relationship;

✚ *Modelli logici:* utilizzati nei DBMS esistenti per l'organizzazione dei dati. I modelli logici sono utilizzati dai programmi ed è a questi che fanno riferimento i programmatori. I più famosi di tali modelli sono quello relazionale (discusso in precedenza), quello reticolare, il gerarchico e quello ad oggetti.

Il modello Entity-Relationship è un modello concettuale dei dati e, come tale, fornisce una serie di strutture, dette "costrutti", atte a descrivere la realtà di interesse in una maniera facile da comprendere e che prescinde dai criteri di organizzazione dei dati nei calcolatori. I costrutti principali del modello sono i seguenti:

✚ *Entità:* classe di oggetti (fatti, cose, persone) dell'applicazione di interesse, con proprietà comuni ed esistenza autonoma;

✚ *Relationship:* legame logico che coinvolge due o più entità;

✚ *Attributo:* proprietà elementare di un'entità o di una relationship, di interesse ai fini dell'applicazione. Associa ad ogni occorrenza di entità o relationship un valore appartenente ad un certo insieme, detto dominio dell'attributo;

- ✚ *Cardinalità di relationship*: coppia di valori associati ad ogni entità che partecipa ad una relationship. Specificano il numero minimo e massimo di occorrenze delle relationship cui ciascuna occorrenza di un'entità può partecipare. Per semplicità, si usano solitamente solo tre simboli: “0” (che indica partecipazione opzionale), “1” (che indica partecipazione obbligatoria) ed “n” (che non specifica alcun limite). Osservando le cardinalità massime relative alle entità coinvolte nella relationship, è possibile identificare tre casi distinti: relationship “uno a uno”, relationship “uno a molti” e relationship “molti a molti”;
- ✚ *Cardinalità di attributi*: è possibile associare cardinalità anche agli attributi, al fine di indicare opzionalità o possibilità di inserire più di un valore;
- ✚ *Identificatore di una entità*: viene specificato per ciascuna entità di uno schema e serve a descrivere i concetti (attributi e/o entità) dello schema che permettono di identificare in maniera univoca le occorrenze delle entità stessa. Un identificatore viene detto

Parte I.5 – “Data Warehousing”

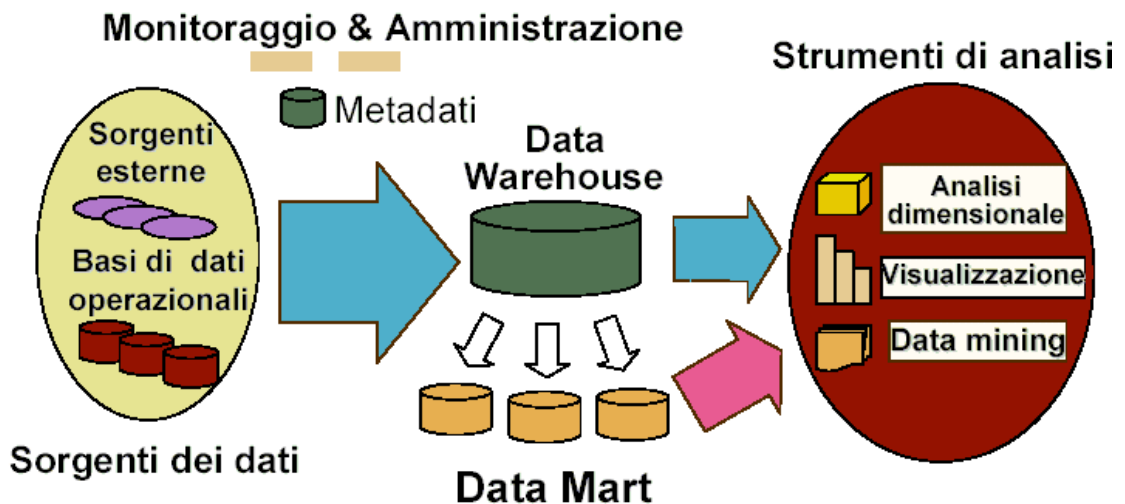
I sistemi informativi/informatici aziendali devono generalmente essere in grado di rispondere a due categorie di richieste, quelle relative alla gestione operativa e quelle inerenti la gestione strategica. Queste due tipologie di richieste fanno capo ad altrettante categorie di elaborazioni, così definite:

- ✚ *On-Line Transaction Processing (OLTP)*: tradizionale elaborazione di transazioni che realizzano i processi operativi dell'azienda-ente;
- ✚ *On-Line Analytical Processing (OLAP)*: elaborazione di operazioni per il supporto alle decisioni.

La configurazione di sistemi informatici destinati a svolgere solo una delle due elaborazioni appena elencate è relativamente semplice. Molto più complesso è invece far convivere i due carichi di lavoro, in virtù soprattutto della diversa tipologia di dati richiesti. I sistemi OLAP si basano su alcuni semplici concetti:

- ✚ *Data Warehouse*: base di dati utilizzata principalmente per il supporto alle decisioni direzionali, integrata aziendale (quindi non dipartimentale), orientata ai dati (e non alle applicazioni), con dati storici ed un ampio orizzonte temporale, non volatile e mantenuta separatamente dalle base di dati di tipo operativo;
- ✚ *Data Warehousing*: processo di integrazione di basi di dati indipendenti in un singolo repository (la Data Warehouse), sul quale sia possibile, facilmente ed efficacemente, eseguire query, generare reports ed effettuare analisi;
- ✚ *Data Mart*: una Data Warehouse integra spesso diversi Data Mart, ciascuno dei quali riguarda un particolare aspetto della realtà aziendale. Solitamente, gli utenti si rivolgono ad un particolare Data Mart.

L'architettura per il data warehousing è schematizzabile in questa figura:



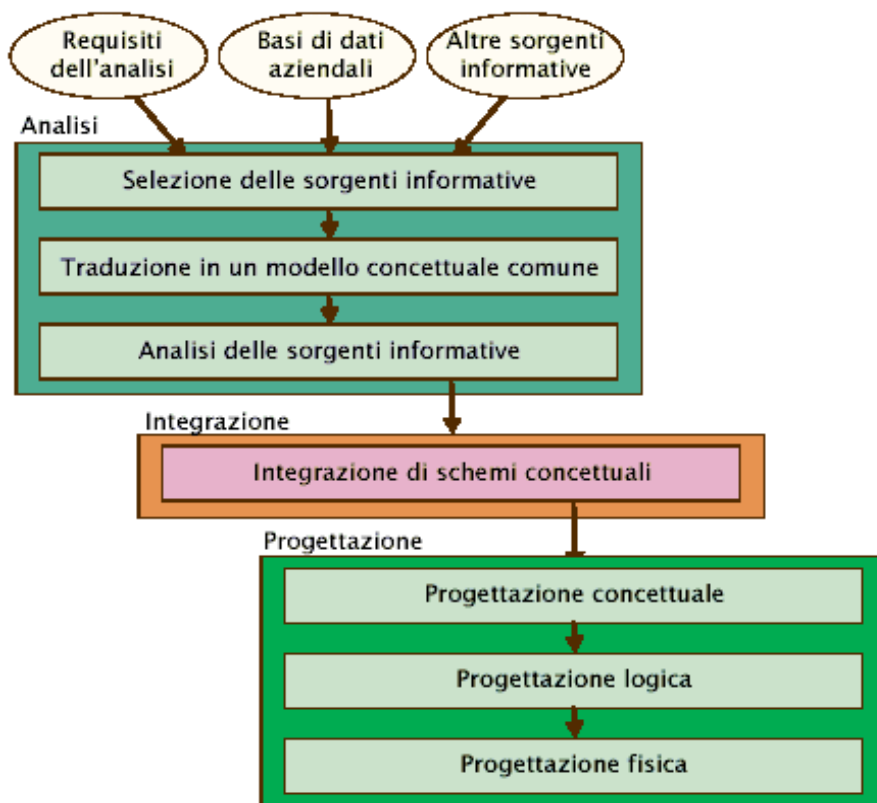
L'analisi dei dati contenuti nelle Data Warehouse e nei Data Mart, avviene rappresentando i dati stessi in una particolare forma multidimensionale, dove i concetti rilevanti sono:

- ✚ *Fatti*: i concetti sui quali centrare l'analisi (ad esempio, per una catena di negozi, il fatto potrebbe essere "vendita");
- ✚ *Misure*: le proprietà atomiche del fatto da analizzare (ad esempio, sempre facendo riferimento alla catena di negozi, "quantità venduta" e "incasso");
- ✚ *Dimensioni*: descrivono la prospettiva lungo la quale effettuare l'analisi (ad esempio, "prodotto", "tempo" e "zona").

Sui dati multidimensionali è possibile effettuare quattro tipi fondamentali di operazioni:

- ✚ *Roll-up*: aggrega i dati ad un livello meno dettagliato;
- ✚ *Drill-down*: disaggrega i dati ad un livello più dettagliato;
- ✚ *Slice & Dice*: seleziona e proietta i dati di interesse;
- ✚ *Pivot*: re-orienta il cubo, solitamente utilizzato come strumento di rappresentazione grafica tridimensionale di fatti, misure e dimensioni.

La progettazione di Data Warehouse è un processo differente rispetto alla progettazione di una base di dati operativa e segue lo schema riportato nella figura cui sotto:



I dati in ingresso corrispondono ai requisiti (cioè alle esigenze aziendali di analisi), alla descrizione delle basi di dati esistenti (con una documentazione sufficiente per la comprensione) ed alla descrizione di altre sorgenti informative (che possono ad esempio essere dati ISTAT, ma comunque dati non di proprietà dell'azienda). Queste sorgenti informative vengono inizialmente filtrate, quindi tradotte in un modello concettuale di riferimento ed infine integrate in un'unica base di dati globale, che rappresenta l'intero patrimonio informativo aziendale. Si arriva così alla fase di progettazione vera e propria, divisa in progettazione concettuale (il cui output è costituito da "schemi di fatto", necessari per rappresentare i concetti rilevanti del modello dimensionale, cioè fatti, misure, dimensioni e gerarchie), logica (il cui output è lo schema logico) e fisica (il cui output è lo schema fisico). La funzione fondamentale della progettazione concettuale è quella di ristrutturare lo schema concettuale in input, traducendo i fatti in entità, individuando nuove dimensioni e raffinando i livelli di ogni dimensione. Per quanto riguarda la progettazione logica, invece, esistono vari modelli logici multidimensionali, i più importanti dei quali sono il modello a stella e quello a fiocco di neve (snowflake), entrambi direttamente implementabili su DBMS relazionali. E' bene sottolineare che i sistemi relazionali per Data Warehouse vengono generalmente chiamati ROLAP (Relational OLAP) e si contrappongono ai sistemi multidimensionali, denominati MOLAP (multidimensional OLAP).