

Database (Base di dati)

Cos'è un database

Per comprendere appieno cos'è un database e quali sono i vantaggi legati al suo impiego, è necessario definire in modo esatto e preciso cosa si intende per:

- Database
- DBMS (Database Management System).

Definizione di database:

Un Database può essere definito come un insieme di informazioni strettamente correlate e memorizzate su un supporto di memoria di massa, costituenti un tutt'uno, che possono essere manipolate da più programmi applicativi.

Il data base, pertanto, è costituito da dati.

Finora abbiamo concepito i dati come file, record, campi, registrazioni su disco, ecc. Adesso interpretiamo questi concetti in modo differente.

Se prima il file era visto come l'unità fondamentale dell'elaborazione dei dati, adesso questo tipo di raggruppamento perde importanza a favore della base di dati, ovvero della **collezione globale** dei dati inerenti un certo soggetto.

Esempio:

Si pensi ad un'agenzia bancaria. L'insieme dei dati relativi ai clienti, alle transazioni, ai conti, ai dipendenti, ecc. costituiscono la base dati. Ciò significa che anche se le entità dei dati sono differenti tra loro (dati che riguardano i dipendenti sono di altro genere, ad esempio, dai dati che riguardano i clienti o i conti bancari) essi comunque appartengono alla medesima realtà, ovvero alla realtà di quella particolare agenzia bancaria.

Definizione di DBMS (Database Management System):

Un DBMS è un sistema software per la gestione di basi dati; esso si occupa dell'aggiornamento, della manutenzione e della consultazione di un insieme di registrazioni contenute in un supporto di memoria di massa.

Il DBMS, pertanto, è un insieme di programmi, che sono rivolti alla gestione di dati memorizzati in archivi. Ovviamente, tra Database e DBMS esiste una forte iterazione per cui spesso si tende a confonderli, ma sono comunque due cose ben diverse e distinte.

Esempio:

Si pensi ad un bambino che sta giocando con delle conchiglie disponendole in un certo ordine, le conchiglie (entità che vengono manipolate) rappresentano il database, il bambino (colui che opera) rappresenta il DBMS.

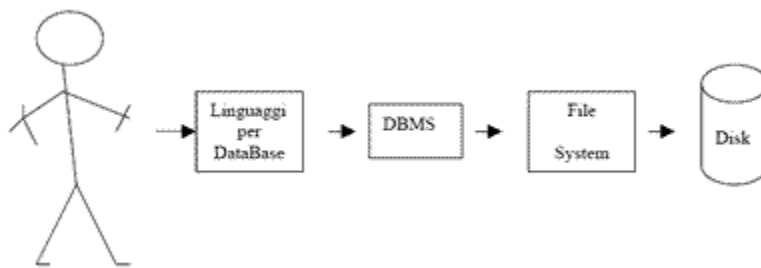
DBMS (Database Management System)

Differenze tra DBMS e file system

Il file system è un nucleo, costituito da programmi, presente in ogni sistema operativo. La sua funzione è quella di gestire le varie operazioni sui file; questa gestione non è visibile all'utente, infatti il file system opera direttamente al servizio di altri programmi o utility.

Il DBMS è per l'appunto un software che poggiando sul sistema operativo utilizza il file system di quest'ultimo.

In questo caso dunque, la maggiore 'distanza' del DBMS dall'hardware, rispetto al file system, permette un grado di iterazione maggiore. Questo significa che l'utente (programmatore, amministratore di sistema, ecc.) non dovrà più avere a che fare con record e file, bensì con entità astratte che rappresentano la realtà.



Componenti di un DBMS

È evidente che un DBMS è un insieme di programmi atti ad effettuare interventi su un database.

Vediamo quali sono le modalità in cui un DBMS può ricevere comandi:

- Direttamente dall'utente in modo interattivo, tramite particolari comandi appartenenti ai linguaggi 'accettati' da quel particolare DBMS.
- Attraverso un programma scritto completamente in uno o più di quei linguaggi di cui parlavamo sopra.
- Tramite un programma scritto in un linguaggio algoritmico tradizionale (COBOL, Pascal, C, ecc.) che ingloba alcuni comandi appartenenti ai linguaggi 'accettati' dal DBMS.

Vediamo ora come possono essere raggruppati questi linguaggi in base alle loro funzioni:

- DDL (*Data Description Language*), tramite i quali si definiscono le strutture del database. Si dice cioè come dovrà essere la base di dati.
- DML (*Data Manipulation Language*), che servono per impartire comandi di elaborazione dei dati.
- QL (*Query Language*) o linguaggi di interrogazione, che presentano una natura interattiva.

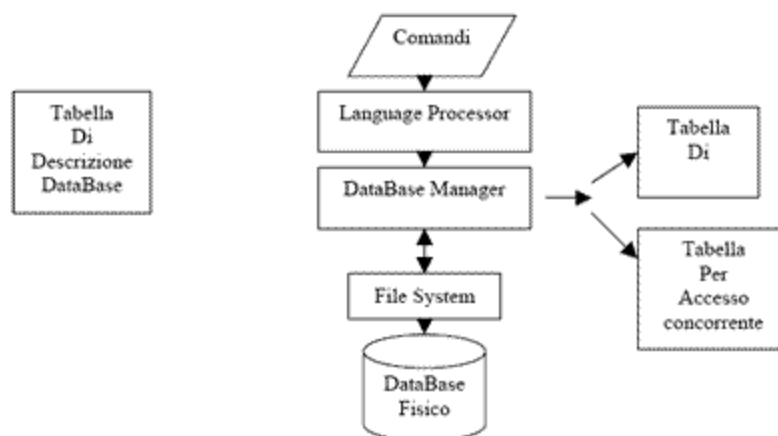
Così, la descrizione delle strutture avverrà tramite il DDL, i programmi di elaborazione saranno scritti in DML e le interrogazioni saranno effettuate con un QL.

Se si desidera fare uso di un linguaggio algoritmico tradizionale, è possibile ricorrere all'*incapsulamento*, ovvero all'inserimento di blocchi di istruzioni scritte in uno dei linguaggi che il DBMS può leggere.

Non sempre la distinzione tra i diversi tipi di linguaggi (DDL, DML, QL) è netta e chiara: vi possono essere infatti linguaggi che comprendono addirittura tutti e tre i tipi di comandi.

Componenti funzionali di un DBMS

- *Tabella di descrizione database*, è una tabella in cui è descritto il modello e le caratteristiche del database.
- *Tabella delle autorizzazioni*, è una tabella in cui sono presenti le informazioni dei vari utenti riguardo ai loro permessi di accesso ai vari dati.
- *Tabella per accesso concorrente*, ha la funzione di permettere la gestione del traffico di più richieste operati contemporaneamente sui medesimi dati.
- *Language processor*, i comandi vengono ricevuti da questa unità, che ha il compito di metterli in relazione con le specifiche di definizione del modello (contenute nella tabella di descrizione del database) e trasmetterli al Database Manager
- *Database Manager*, ha il ruolo di ricevere i comandi espressi a livello concettuale cioè operanti sul modello astratto dei dati e tradurli in comandi a livello fisico trasmettendoli al file system del sistema operativo non prima di aver fatto gli opportuni controlli sulla *tabella di autorizzazione* e sulla *tabella di accesso concorrente*.



Livelli di astrazione di un DBMS (introduzione)

Abbiamo accennato al fatto che il DBMS ha, tra le sue funzioni, quella di mantenere un modello astratto dei dati che consenta all'utente di concepire le informazioni secondo strutture riferite direttamente alle entità reali che i dati sono preposti a rappresentare.

Abbiamo inoltre visto che spetta al DBMS, tramite il Database Manager, tradurre le operazioni ordinate dall'utente sulle strutture astratte in operazioni sui file.

Abbiamo visto che l'attività del DBMS consiste nel ricevere comandi, metterli in relazione con il modello dei dati, verificare l'eseguibilità e tradurli in operazioni sugli archivi.

Nell'esecuzione di tali operazioni, il DBMS, così come ogni altro programma applicativo, dipende dal sistema operativo: per l'utilizzo concorrente delle risorse, per l'input/output a video o su stampa, per gli interventi in memoria di massa e così via.

Le funzioni del DBMS

È sufficiente pensare al significato dell'acronimo DBMS (Database Management System ovvero Sistema gestore della base dei dati) per capire che la principale funzione di un DBMS è quella di governare ogni operazione di accesso al Database, sia per quanto riguarda aggiornamenti o ricerche effettuati sugli archivi, sia per le richieste da parte di più applicazioni contemporaneamente.

La caratteristica principale del DBMS consiste nel mantenimento di un **modello di dati**, ovvero di un'astrazione logica tramite la quale l'utente può vedere i **dati** non in termini di record, file, campi,

bensì in termini di **unità informative**, direttamente riferibili agli oggetti della realtà descritti dall'informazione stessa.

Esempio:

Prendiamo in considerazione l'entità *dipendenti* di una ditta; è possibile mettere in relazione (il concetto relazione nasce dalla matematica, ed in particolare dalla teoria degli insiemi) i dati anagrafici ad essi relativi e disegnare la seguente:

DIPENDENTI (Cognome, Nome, Data_nascita)

Questa relazione può corrispondere ad un file, ma ciò non è necessario: non è detto vi sia corrispondenza tra *relazioni*, ovvero tra entità costituite da informazioni unitarie, e *registrazioni*, ovvero record di dati memorizzati su disco.

Tramite il modello dei dati, è possibile progettare e gestire un database in maniera completamente svincolata:

- Dalle applicazioni
- Dall'organizzazione fisica dei dati

Esempio:

Supponiamo di volere stampare l'elenco dei dipendenti della ditta che hanno uno stipendio base superiore ad un certo ammontare.

Operando tramite il DBMS è possibile mettere in relazione tra loro i dati anagrafici dei dipendenti (in questo caso il nominativo) con dati di tipo amministrativo (la paga base), registrati dal punto di vista fisico, in due archivi differenti. L'utente, tramite un'astrazione dei dati stessi, può combinare i valori ed operare solo sulla porzione di quelli che gli interessano; nel caso in esame opererà su una relazione relativa ai dipendenti con paga superiore all'ammontare indicato, effettuandone la stampa.

Il linguaggio di interrogazione, o *query language*, fornisce inoltre il grosso vantaggio di permettere una consultazione interattiva della base di dati, senza dover redigere un programma apposito per ciascuna esigenza informativa. All'utente basta impostare il comando adeguato per ottenere la risposta cercata.

Esempio:

Utilizzando la programmazione tradizionale, ciascuna richiesta informativa necessita di un programma: i dipendenti con un certo numero di anni di anzianità, la lista di coloro che hanno figli, l'elenco dei lavoratori in ferie in un determinato periodo, ecc. È necessario impostare varie routine con diverso accesso ai dati, che verificano condizioni ogni volta differenti.

Tramite il query language, invece, ciascuna esigenza informativa è soddisfatta tramite un'interrogazione: un solo comando va a sostituire un intero programma di centinaia di righe.

Accanto a tali funzioni, il DBMS ne espleta anche altre relative per lo più alla *sicurezza* (non permettere all'accesso ai dati a persone non autorizzate) e all'*integrità* dei dati (garantire e permettere delle modifiche e aggiornamenti che non stravolgano lo schema logico della base di dati).

Vediamo quali sono queste altre funzioni:

- *La gestione delle transazioni* (transazione: è un'azione che opera sui dati del database: li legge, li scrive, ecc.), per fornire un accesso corretto e concorrente al data base da parte di molti utenti contemporaneamente.
- *L'accesso controllato*, per limitare l'accesso ai dati nei riguardi degli utenti non autorizzati e controllare la validità dei dati immessi.
- *La capacità di recupero*, cioè la possibilità di ripristino a seguito di guasti del sistema senza perdere i dati.

Livelli di astrazione di un DBMS

Abbiamo accennato al fatto che il DBMS ha tra le sue funzioni, quella di mantenere un modello astratto dei dati che consenta all'utente di concepire le informazioni secondo strutture riferite direttamente alle entità reali che i dati sono preposti a rappresentare.

Abbiamo inoltre visto che spetta sempre al DBMS, per mezzo del Database Manager, tradurre le operazioni ordinate dall'utente sulle strutture astratte in operazioni sui file.

Vediamo adesso quali sono i livelli di astrazione consentiti.

Livello fisico

Il livello fisico del database è rappresentato dalle strutture di memoria di massa usate per conservare i dati e per accedervi in modo rapido ed efficiente.

È necessario distinguere tra:

- I dati veri e propri
- Le strutture che li contengono e che ci permettono di accedere ai medesimi

Il livello fisico è, per l'utente, del tutto trasparente: egli, infatti, non si preoccupa affatto di come i dati vengano registrati sui supporti, tale funzione è compito esclusivo del DBMS. L'utente si occuperà principalmente del cosa vi è registrato: quali sono i dati e in quale relazione si trovano tra loro.

Essendo la gestione dei file realizzata dal sistema, l'utente non deve interessarsi della forma assunta dai dati in memoria di massa; non gli resta dunque che occuparsi del valore informativo dei dati.

Livello concettuale

Il livello concettuale rappresenta la struttura globale del database, relativa a tutte le informazioni in esso presenti, rappresentate in modo organizzato tramite un modello astratto.

La descrizione è realizzata tramite il Data Description Language (DDL), ovvero il linguaggio di descrizione dei dati, che fornisce la descrizione del modello astratto del database.

Il modello è composto da *entità*, a ciascuna entità saranno riferite informazioni, rappresentate da *attributi* inerenti all'entità medesima. Si crea quindi un rapporto biunivoco tra un'entità del mondo reale, che costituisce l'oggetto di riferimento, e un insieme di dati omogenei e coerenti all'interno del database. Questi dati riporteranno le caratteristiche dell'oggetto a cui si riferiscono, realizzando per l'appunto un processo di *astrazione* dal livello reale al livello concettuale.

Un'entità è costituita da ciascun *oggetto* al quale si vogliono riferire informazioni: sono entità gli studenti di una scuola, i dipendenti di un'azienda, i correntisti di un istituto di credito, i libri di una biblioteca. A ciascuna entità fanno riferimento dati ben precisi: nome, data di nascita, reparto, classe, titolo, autore, ecc. Gli attributi sono collegati alle entità, costituendo un oggetto di potenziale informativo maggiore rispetto ad una collezione di informazioni non correlate.

Esempio:

Prendiamo in esame l'entità STUDENTI

STUDENTI (nome, classe, sezione)

Il dominio di <nome> sarà costituito da una stringa di caratteri, il dominio <classe> da un intero compreso tra 1 e 5, il dominio di <sezione> da un carattere alfabetico limitato alle sole sezioni presenti nella scuola.

Livello esterno

Una volta definito lo schema concettuale, è possibile limitare l'accesso ai dati ad alcuni utenti. Quindi sono realizzabili accessi e navigazioni personalizzate.

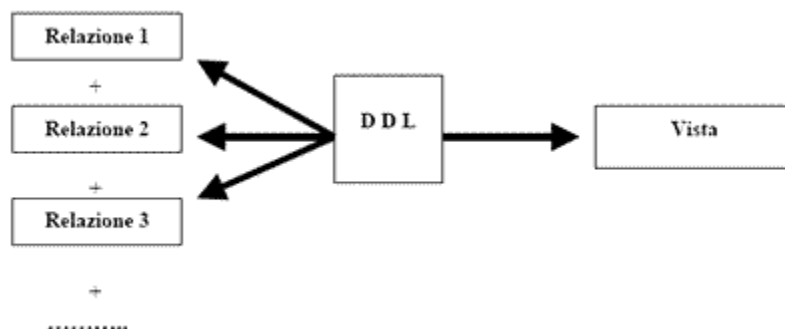
Ciò è reso possibile dalle funzioni del DBMS relative al *livello* esterno che danno la sensazione a ciascuno di essere l'unico utilizzatore dei dati che vede.

Accesi e navigazioni si attuano attraverso schemi detti *viste logiche* dei dati.

Definizione di vista:

Una **vista** è l'**astrazione** di una parte del data base concettuale che coinvolge i dati dell'istanza del data base limitatamente alla **porzione interessata**.

Attraverso il DDL, l'utente può descrivere viste logiche dei dati che abbiano tra loro informazioni appartenenti a entità diverse, creando così *nuove entità informative*.



Si può inoltre limitare l'accesso in *orizzontale*, cioè a una sola parte degli attributi di una entità, oppure in *verticale*, condizionando il dominio di uno o più attributi. In questo modo, si verrà a disporre di una nuova configurazione della base dei dati, rispondente in modo completo alle esigenze di quel utente.

Esempio:

Tornando alla entità di cui sopra:

STUDENTI (nome, classe, sezione)

Qualora ci interessi una stampa dei nomi, dei soli studenti che frequentano la classe quinta, è possibile descrivere una vista logica dei dati limitando il dominio di <classe> al valore 5 e gli attributi a <nome>. Se invece si desidera il listato dei soli alunni della sezione E, si limiterà il dominio di <sezione> a E appunto.

In sostanza, la vista logica è un modo di vedere, accedere e navigare sui dati che sono stati descritti in precedenza dallo schema concettuale e tramite questo registrati nell'istanza del data base a livello fisico.

A questo punto siamo in grado di riassumere quanto detto per i tre livelli:

- *Il livello fisico* riguarda l'effettiva memorizzazione dei dati, organizzati in file, record e strutture di accesso.
- *Il livello concettuale* riguarda la struttura logica assunta dai dati registrati, quindi il loro schema astratto.
- *Il livello esterno* si riferisce al modo in cui ciascun utente può vedere gli stessi dati, che mantengono l'organizzazione fisica e concettuale precedentemente descritte, ma vengono messe a disposizione secondo il formato richiesto.

Il concetto fondamentale da comprendere è la differenza tra i *dati* e lo *schema*; solo i file contengono effettivamente raggruppamenti di byte, mentre gli schemi, le entità e le viste non fanno altro che descrivere il formato con cui possono essere di volta in volta manipolati. Tale realtà, tuttavia, è completamente trasparente all'utente, che opera tramite proprie viste logiche come se si trattasse di una base dati fisica vera e propria.

Perché utilizzare un DBMS

Il largo successo ottenuto nel corso degli ultimi quindici anni dai sistemi di gestione delle basi di dati lascia intendere che i vantaggi legati al loro utilizzo siano numerosi. I DBMS sono sistemi di utilità che si occupano di una vasta gamma di operazioni relative alla gestione dei dati non volatili del sistema di elaborazione; in particolare si tratta di operazioni che:

- *Sostituiscono il programmatore*: in questo caso, i vantaggi sono legati al fatto che il sistema esegue funzioni che in precedenza erano a carico dei programmatori. Si pensi alla possibilità di consultare la base di dati senza necessità di redigere un apposito programma, ma attraverso un semplice comando di interrogazione, utilizzando il query language; in questo caso l'accesso ai dati è realizzato tramite routine del DBMS.
- *Arricchiscono la gestione precedente*, ovvero permettono l'esecuzione di nuove funzioni, in basi di dati già esistenti, aumentandone quindi il potenziale informativo senza che sia necessario aumentare la quantità di dati raccolti e memorizzati.

In una gestione tradizionale è conveniente scrivere un programma di interrogazione solo quando questa esigenza interessa più utenti e si verifica con una certa frequenza. Con i database invece ciascun utente può ottenere informazioni di grande valore semplicemente manipolando i dati secondo i criteri che desidera.

In sostanza, con il DBMS si realizza una gestione centralizzata e controllata della base dati i cui vantaggi principali possono essere così sintetizzati:

- *Riduzione della ridondanza dei dati*, dove con ridondanza si intende la possibilità che i dati si presentino ripetutamente nella base.
- *Eliminazione dell'incongruenza*.
- *Condivisione dei dati* da parte di tutte le applicazioni che ne facciano richiesta.
- *Sicurezza e riservatezza* delle informazioni, per ridurre il rischio di distruzione di dati e di accessi non autorizzati.
- *Ottimizzazione della struttura* della base dati, che ne facilita l'accesso e la manutenzione e ne garantisce una crescita ordinata.
- *Indipendenza dei dati dalle applicazioni*.

Modelli logici

Il modello logico discende dal modello concettuale e disegna un'architettura che tiene conto delle strutture proprie di quel particolare tipo di database. Ciò significa che è possibile realizzare diversi tipi di database a partire da uno stesso modello concettuale.

Nel corso degli anni si sono sviluppati vari tipi di modelli logici, tutti possono essere ricondotti a tre categorie principali: **gerarchici**, **reticolari** e **relazionali**.

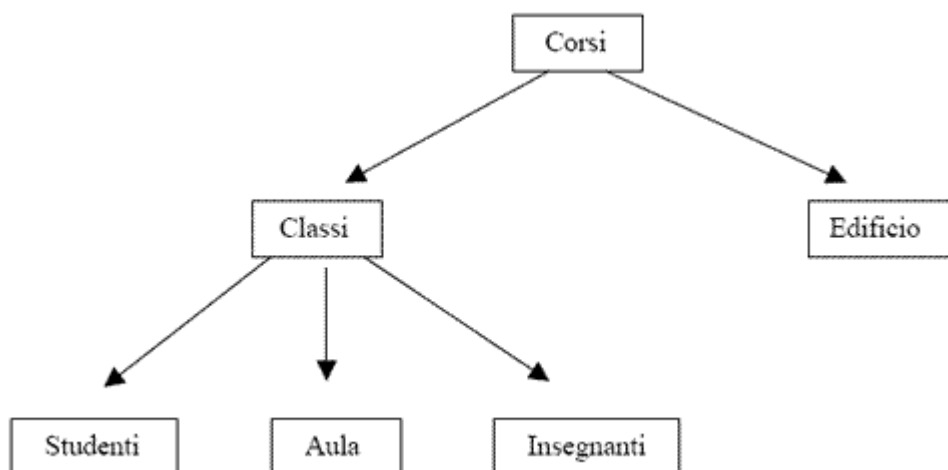
Modello gerarchico:

Volendo tracciare un percorso storico che attraversi l'evoluzione subita dai DBMS nel corso degli anni, è necessario iniziare la trattazione a partire dal *modello gerarchico*. Si può fissare la data di nascita di questo modello alla fine degli anni '60, quando IBM sviluppa e introduce sul mercato IMS, il primo database gerarchico, ma anche il primo DBMS in assoluto.

Definizione:

Un database gerarchico è un insieme di archivi. Gli archivi sono composti da record chiamati *segmenti*. I segmenti sono in rapporto gerarchico tra loro attraverso legami di tipo *padre-figlio*.

La struttura ad albero che caratterizza il modello gerarchico si basa sulla possibilità di individuare un *segmento principale*, il padre o la radice, dal quale dipendono n *segmenti figli*, che a loro volta si trasformano in padri per altri figli e così via. **A questi, in virtù della totale dipendenza dal padre, è possibile fare riferimento solo attraverso il passaggio dal nodo principale.** Non è possibile dal figlio risalire al padre.



Questa architettura mal si adatta ad una gestione moderna e dinamica delle basi di dati.

Modello reticolare:

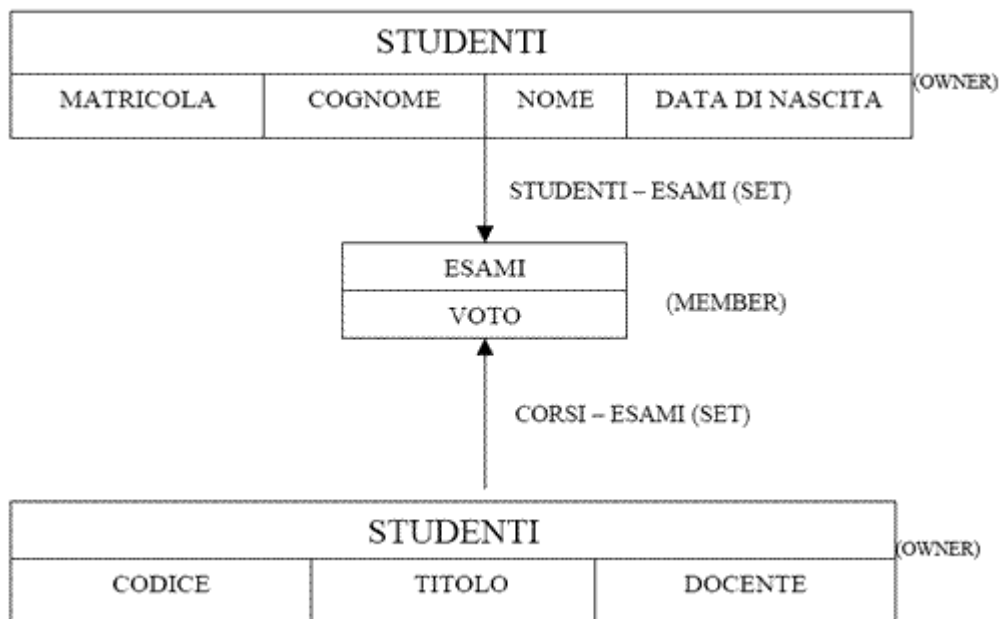
Il modello gerarchico rappresenta una prima soluzione al problema della gestione di grosse moli di dati ma la sua intrinseca rigidità ne limita la potenzialità; per questo, nasce il *modello reticolare* che dotato di maggiore flessibilità, può adattarsi a situazioni più complesse.

Il modello reticolare può essere visto come un'estensione del modello gerarchico, al quale siano apportati importanti miglioramenti :

In una struttura gerarchica un segmento figlio può avere solo un segmento padre; non è così nel modello reticolare: ogni record può avere un numero qualsiasi di record subordinati e di record precedenti e le correlazioni vengono espresse attraverso record particolari, chiamati *record di collegamento* (member), che formano delle catene tra le varie parti del sistema.

La struttura:

le strutture utilizzate nel modello reticolare sono due, il *record* (si pensi ai comuni file) e il set, che permette di correlare i record, per mezzo di catene di puntatori. Dunque una base di dati reticolare è definita con riferimento ad uno schema, che contiene tipi record collegati fra loro da tipo set.



Il modello relazionale

Il modello relazionale, fu proposto per la prima volta, nel 1970. Fin da allora ha avuto un crescente successo, dovuto principalmente alla sua semplicità e alla sua flessibilità.

Su cosa si basa il modello relazionale:

Il *modello relazionale* si basa su due concetti *relazione* e *tabella*. La nozione di relazione proviene dalla teoria degli insiemi, mentre il concetto di tabella è semplice ed intuitivo.

La struttura del modello relazionale è molto semplice e potente. Al tempo stesso, essa impone però un certo grado di rigidità. In ogni relazione possiamo rappresentare solo tuple corrispondenti allo schema della relazione stessa. In effetti in molti casi, i dati disponibili possono non corrispondere esattamente al formato previsto. Ad esempio:

Persone (Cognome, Nome, Indirizzo, Telefono)

Il valore dell'attributo Telefono potrebbe non essere disponibile per tutte le tuple. Vale la pena notare che non sarebbe corretto utilizzare un valore del dominio per rappresentare l'assenza di informazione, in quanto in tal modo si potrebbe ingenerare confusione.

Per rappresentare in modo semplice la non disponibilità di valori, viene assunto un particolare valore per quel attributo detto *valore nullo*.

In genere quando si definisce una relazione bisogna specificare quali saranno gli attributi in cui è possibile inserire il valore nullo.

Vincoli di integrità:

In una base di dati, è opportuno evitare il più possibile che vengano inseriti dati privi di senso o sbagliati. Per limitare che ciò accada, durante la creazione del database vengono stabiliti dei vincoli.

Un **vincolo** è 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 o lecite le istanze che soddisfano tutti i vincoli.

Esempi di vincoli di integrità:

Non permettere l'inserimento del codice di avviamento postale usando lettere invece di cifre numeriche o non accettare che vengano inseriti più di 5 cifre.

Non consentire la cancellazione di un capo vendite senza che i suoi subalterni (venditori) siano gerarchizzati ad altro capo vendita o cancellati precedentemente.

Concetto di chiave:

- Si definisce **chiave candidata** l'attributo o l'insieme di attributi che permettono di individuare univocamente la tupla all'interno della relazione.
- Si definisce **chiave primaria** la chiave con il minor numero di attributi.
- Si definisce **chiave esterna** l'attributo, o l'insieme di attributi, che può essere usata come chiave primaria per un'altra relazione.

AUTO

Targa	Proprietario	Indirizzo
RM 1A2396	Verdi Piero	Via Tigli
FI 1A2300	Verdi Piero	Via Tigli
TV 2F4560	Bianchi Antonio	Via Tigli

Possono essere considerate chiavi candidate tutti e tre gli attributi insieme della tabella Auto, o gli attributi Targa e Proprietario insieme. Non possono essere considerati chiave candidata gli attributi Proprietario e Indirizzo, sia da soli che insieme. È decisamente chiave primaria l'attributo Targa. L'attributo Proprietario comparando come chiave primaria nella relazione Proprietari Auto (Persona) è chiave esterna.

PROPRIETARI AUTO

Persona	Indirizzo
Verdi Piero	Via Tigli
Bianchi Antonio	Via Tigli

Normalizzazione dei database

Per eliminare definitivamente la ridondanza dalle relazioni e quindi dal database, si ricorre ad un processo di normalizzazione: si tratta di un procedimento di tipo graduale, che realizza un'ottimizzazione progressiva a partire da relazioni non normalizzate fino a raggiungere un certo livello di normalizzazione.

Prima di entrare in merito alla normalizzazione è bene introdurre alcuni particolari concetti:

Definizione di ridondanza:

Si ha ridondanza dei dati ogni volta che vengono memorizzati inutilmente dei dati ripetuti.

Esempio:

Si supponga di disporre di una tabella in cui vengono registrati tutti gli acquisti di merci effettuati da un'impresa mercantile.

Acquisti (Cod_fornitore, Nome_fornitore, Codice_fiscale, Merce, Quantità)

In un caso come questo i dati di Fornitore e Codice_fiscale vengono ripetuti inutilmente ogni volta che compare la registrazione di un acquisto. Si può ovviare ad una tale problema separando le informazioni ridondanti della tabella acquisti:

Acquisti (Merce, Cod_fornitore, Quantità)

Fornitori (Cod_fornitore, Nome_fornitore, Codice_fiscale)

In questo modo se gli acquisti effettuati da un certo fornitore compaiono mille volte, non dovremo per mille volte ripetere inutilmente l'informazione Nome_fornitore, Codice_fiscale.

Dipendenza funzionale:

Si ha dipendenza funzionale tra un attributo x e una chiave se i valori di x dipendono dai valori assunti dalla chiave, ovvero se si viene a determinare una relazione biunivoca tra i valori di x e i valori della chiave.

Esempio:

Stabiliamo che nella relazione di esempio abbiamo come chiave primaria Nome e Cognome.

ANAGRAFICA (Nome, Cognome, Indirizzo)

È evidente che Indirizzo è dipendente funzionalmente dalla chiave, ciò significa che per ogni persona corrisponde un indirizzo.

Linguaggio di interrogazione di un DBMS: SQL

SQL non identifica un prodotto commerciale, ma un linguaggio nello stesso modo in cui C e Basic indicano linguaggi generali e non compilatori particolari.

È un linguaggio, che serve per eseguire varie operazioni sia sui dati che sulle strutture che li contengono. La sigla, acronimo di *Structured Query Language*, è ormai diventata sinonimo di linguaggio standard per la gestione dei database relazionali.

SQL è dunque un linguaggio per la gestione di database relazionali, quindi assolve alle funzioni di Data Description Language (linguaggio di descrizione dei dati e delle strutture che li conterranno), di Data Manager Language (linguaggio per la manipolazione dei dati) e di linguaggio di interrogazione.

Il termine SQL può generare confusione. La lettera S, iniziale di Structured (Strutturato), e la lettera L, iniziale di Language, sono abbastanza semplici, ma la lettera Q si presta a varie interpretazioni. Ovviamente Q sta per Query che se fosse interpretata alla lettera, limiterebbe il linguaggio SQL a uno strumento per interrogare il database. In effetti SQL fa molto di più che porre delle domande.

Gli informatici chiamano questo linguaggio di alto livello o dichiarativo perché permette di svolgere operazioni dichiarando **cosa** si deve ottenere e non **come** si deve ottenere. Ricordiamo che i linguaggi di terza generazione o procedurali sono quelli dove bisogna specificare il *come si fa*, non è sufficiente dichiarare il *cosa si deve fare*. Non è così per questo linguaggio, che pur limitando le scelte del programmatore e l'efficienza del programma, libera lo sviluppatore dal gravoso compito di scrivere pagine e pagine di codice. Chi usa questo linguaggio però, non è solo chi programma, ma anche chi si avvicina all'informatica marginalmente e per riflesso. Queste persone sono gli impiegati, i professionisti, i commessi, i magazzinieri, ecc. insomma chiunque ha la necessità di manipolare o consultare basi di dati. Forse la causa del suo grande successo sta nella sua semplicità di utilizzo. Non bisogna, però farsi ingannare, perché se da un lato SQL è intuitivo e semplice, da un altro, per essere capito a fondo richiede di essere studiato con attenzione per capirne tutte le sfumature e le notevoli potenzialità.

Storia di SQL

Le origini di SQL risalgono all'inizio degli anni 70 in California, quando la società IBM sviluppa il System R, un applicativo per la gestione dei dati, il cui linguaggio veniva chiamato *Sequel*. Questo linguaggio rappresentava l'embrione di quello che sarebbe poi diventato l'attuale SQL. Infatti alla fine degli anni 70, sempre l'IBM, sviluppa un altro prodotto il DB2 (un sistema per la gestione di database relazionali o RDBMS, *Relational Database Management System*) che utilizza una primordiale versione di SQL. Da allora si sono succeduti un gran numero di prodotti che implementano questo linguaggio e ogni produttore, aggiungendo delle variazioni e estensioni proprie, ha contribuito alla creazione della miriade di dialetti che oggi vengono chiamati SQL.

Standardizzazione di SQL

I due enti che si occupano del processo di standardizzazione ANSI (*American National Standards Organization*) e ISO (*International Standards Organization*), stanno svolgendo, ormai da anni, azioni di promozione dello standard SQL.

Sebbene questi enti preparino le specifiche cui devono adeguarsi i progettisti dei vari DBMS tutti i prodotti che implementano SQL differiscono in maniera più o meno marcata dagli standard ufficiali, aggiungendo delle variazioni alla sintassi o ampliando alcune funzioni.

Nel 1986 è stato promulgato il primo standard; esso possedeva già gran parte delle primitive di formulazione di interrogazioni, ma offriva un limitato supporto per la definizione e manipolazione dei dati e delle strutture logiche che avrebbero dovuto contenerli.

Nel 1989 lo standard è stato ulteriormente esteso ma in modo limitato; l'aggiunta più significativa di questa versione è stata la definizione dell'integrità referenziale. Si fa riferimento a questa versione dello standard usando il nome *SQL-89*.

Nel 1992 è stata pubblicata un'altra versione, in gran parte compatibile con quella precedente ma arricchita da un gran numero di nuove funzionalità. Si fa riferimento a questa ulteriore versione usando i nomi *SQL-92* o *SQL-2*.

L'opera di standardizzazione sta continuando ancora, infatti si sta lavorando ad un'ennesima versione chiamata *SQL-3*, allo stato attuale non ancora pubblicata.