INAIL
ISTITUTO NAZIONALE PER L'ASSICURAZIONE CONTRO GLI INFORTUNI SUL LAVORO

INAIL FORMAZIONE

Materiale Didattico

LE BASI DATI

CONSULENZA PER L'INNOVAZIONE TECNOLOGICA

SERVIZIO FORMAZIONE

Redattori	Roberto Battista

INDICE

PAG.

1. SISTEMI DI GESTIONE

Nello svolgimento di ogni attività sono essenziali la disponibilità di informazioni e la capacità di gestirle in modo efficace; ogni organizzazione è dotata di un sistema informativo che organizza e gestisce le informazioni necessarie per perseguire gli scopi dell'organizzazione stessa.
L'esistenza di un sistema informativo è in parte indipendente dalla sua automatizzazione (basti pensare ai servizi anagrafici che esistono da secoli prima dell'invenzione e della diffusione dei calcolatori); per indicare la porzione automatizzata del sistema informativo viene di solito utilizzato il termine sistema informatico.

I sistemi informatici garantiscono che i dati vengono raccolti, organizzati e conservati in modo permanente su dispositivi per la loro memorizzazione, aggiornati per riflettere rapidamente le loro variazioni e resi accessibili alle interrogazioni degli utenti, talvolta distribuiti in modo capillare sul territorio.
L'attenzione ai dati ha caratterizzato le applicazioni dell'informatica fin dalle sue origini, ma sistemi software specificamente dedicati alla gestione dei dati sono stati realizzati solo a partire dalla fine degli anni sessanta, e tuttora alcune applicazioni non ne fanno uso.
In assenza di un software specifico, la gestione dei dati è affidata ai linguaggi di programmazione tradizionali, ad esempio C e Fortran, oppure, in tempi più recenti, ai linguaggi a oggetti, tra cui C++, Smalltalk e Java.
Molte applicazioni sono scritte in COBOL, un linguaggio di programmazione particolarmente adatto alle "applicazioni gestionali", finalizzate cioè alla gestione dei dati.

L'approccio convenzionale alla gestione dei dati sfrutta la presenza di archivi o file per memorizzare i dati in modo persistente sulla memoria di massa.
Un file consente di memorizzare e ricercare dati, ma fornisce solo semplici meccanismi di accesso e di condivisione.
Secondo questo approccio, le procedure scritte in un linguaggio di programmazione sono completamente autonome; ciascuna di esse definisce e utilizza uno o più file "privati".
Eventuali dati di interesse per più programmi sono replicati tante volte quanti sono i programmi che li utilizzano, con evidente ridondanza e possibilità di incoerenza.
Le basi di dati sono state concepite in buona misura per superare questo tipo di inconvenienti.

Una base di dati (in inglese database) è, nella sua accezione generica, una collezione di dati organizzata per reperire le informazioni necessarie allo svolgimento delle attività di un'organizzazione (azienda, ente, persona, ecc.).

In un'accezione specifica, una base di dati è una collezione di dati gestita da un particolare sistema software che prende il nome di data base management system (DBMS).
Una base di dati, nella sua accezione generica, rientra nella definizione di sistema informativo e, nella sua accezione specifica, rientra nella definizione di sistema informatico.
Un sistema di gestione di basi di dati (DBMS) è un sistema software in grado di gestire basi di dati, cioè collezioni di dati che siano grandi, condivise e persistenti, assicurando la loro affidabilità e privatezza.
Come ogni prodotto informatico, un DBMS deve essere efficiente ed efficace.
Esplicitiamo le caratteristiche dei DBMS e delle basi dati che sono alla base delle definizioni date in precedenza.

- Le basi di dati possono essere grandi, nel senso che possono avere anche dimensioni enormi (migliaia di miliardi di byte) e comunque in generale dimensioni molto maggiori della memoria centrale disponibile. Di conseguenza, i DBMS debbono prevedere una gestione dei dati in memoria secondaria. Ovviamente, possono esistere anche basi di dati "piccole", ma i sistemi debbono poter gestire i dati senza porre limiti alle dimensioni, a parte quelle fisiche dei dispositivi.
- Le basi di dati sono condivise, nel senso che applicazioni e utenti diversi debbono poter accedere, secondo opportune modalità, a dati comuni. È importante notare che in questo modo si riduce la ridondanza dei dati, poiché si evitano ripetizioni e, conseguentemente, si riduce anche la possibilità di inconsistenze: se esistono varie copie degli stessi dati, è possibile che esse non siano allineate; viceversa, se ogni dato è memorizzato sul calcolatore in modo univoco, non è possibile incorrere in disallineamenti. Per garantire l'accesso condiviso ai dati da parte di molti utenti che operano contemporaneamente, il DBMS dispone di un meccanismo apposito, detto controllo di concorrenza.
- Le basi di dati sono persistenti, cioè hanno un tempo di vita che non è limitato a quello delle singole esecuzioni dei programmi che le utilizzano. In contrasto, ricordiamo che i dati gestiti in memoria centrale da un programma hanno una vita che inizia e termina con l'esecuzione del programma; tali dati, quindi, non sono persistenti.
- I DBMS garantiscono la affidabilità, cioè la capacità del sistema di conservare sostanzialmente intatto il contenuto della base di dati (o almeno di permetterne la ricostruzione) in caso di malfunzionamenti hardware e software. A questo scopo i DBMS forniscono specifiche funzionalità di salvataggio e ripristino (backup e recovery).
- I DBMS garantiscono la privatezza dei dati. Ciascun utente, riconosciuto in base ad un nome d'utente, che è specificato

all’atto di interagire con il DBMS, viene abilitato a svolgere solo determinate azioni sui dati, attraverso meccanismi di autorizzazione.

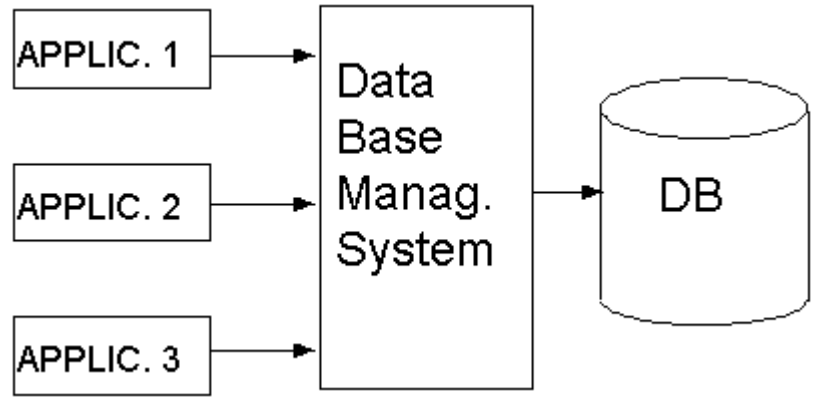
- Per efficienza intendiamo la capacità di svolgere le operazioni utilizzando un insieme di risorse (tempo e spazio) che sia accettabile per gli utenti. Questa caratteristica dipende dalle tecniche utilizzate nell’implementazione del DBMS e dalla bontà di realizzazione della base di dati da parte dei suoi progettisti. Va sottolineato che i DBMS forniscono un insieme piuttosto ampio di funzionalità che richiedono molte risorse, e quindi possono garantire efficienza solo a condizione che il sistema informatico su cui sono installati sia adeguatamente dimensionato.
- Per efficacia intendiamo la capacità della base di dati di rendere produttive, in ogni senso, le attività dei suoi utenti. Questa definizione è chiaramente generica e non corrisponde ad una funzione specifica, dato che un DBMS fornisce vari servizi e funzionalità ad utenti differenti. L’attività di progettazione della base di dati e delle applicazioni che la utilizzano mira essenzialmente a garantire una buona efficacia complessiva del sistema.

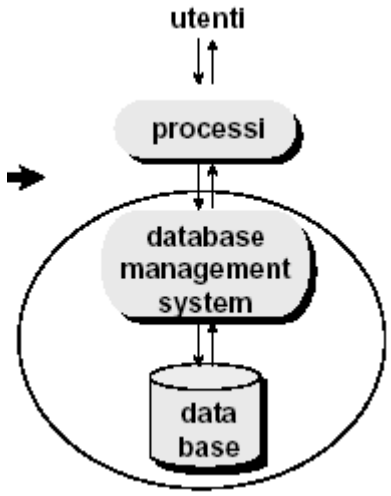
È importante sottolineare che la gestione di collezioni di dati grandi e persistenti è possibile anche per mezzo di strumenti meno sofisticati dei DBMS quali ad esempio i file, presenti in tutti i sistemi operativi.
I file sono stati introdotti per gestire insieme di dati “localmente” ad una specifica procedura o applicazione.

I DBMS sono stati concepiti e realizzati per estendere le funzioni dei file system, fornendo la possibilità di accesso condiviso agli stessi dati da parte di più utenti e applicazioni, e garantendo anche molti altri servizi in maniera integrata.
Precisiamo inoltre che i DBMS utilizzano, a loro volta, dei file per la memorizzazione dei dati; però, i file gestiti dal DBMS ammettono organizzazioni dei dati più sofisticate.

Alcuni esempi di DBMS sul mercato sono: DB2, Oracle, SQLServer, Access.

Applicazioni diverse, DBMS, DB unico





2. MODELLI DEI DATI: GERARCHICO, RETICOLARE, RELAZIONALE

I "protagonisti" di una base di dati sono i dati relativi ad una certa realtà (ad esempio la realtà di un'università, di un'azienda, di un ente pubblico, ecc.). Quindi una base di dati contiene i dati di una certa realtà e permette il reperimento di informazioni che descrivono tale realtà.
Come vengono descritti e rappresentati i dati relativi alla realtà di interesse di una base di dati? Ovvero, come vengono modellizzati i dati di interesse di una base di dati?
In generale, un "modello dei dati" è un insieme di costrutti utilizzati per descrivere e rappresentare i dati propri di una realtà di interesse.
I livelli di modellizzazione dei dati di una determinata realtà di interesse di una base di dati sono tre:

- Livello concettuale
- Livello logico
- Livello fisico

A livello concettuale, più che modellizzare i dati relativi ad una certa realtà, si cerca di descrivere i concetti della realtà stessa (modellizzazione del mondo reale).
I "modelli concettuali" offrono infatti i concetti e i costrutti che permettono di descrivere una realtà di interesse in maniera indipendente dall'organizzazione della base di dati che poi la rappresenterà. Essi sono utilizzati nella fase preliminare di progettazione di una base di dati.
Il modello concettuale più usato è il modello Entità-Relazione (Modello E-R), in cui il concetto di relazione è da intendersi nel senso di legame (in inglese relationship). Il modello E-R descrive la realtà di interesse attraverso i costrutti principali di entità e di relazione

A livello logico, si modellizzano i dati relativi ad una certa realtà al fine di descrivere la loro organizzazione nella base di dati.
I "modelli logici" offrono i concetti e i costrutti che permettono di organizzare i dati nella base di dati in modo indipendente dalle strutture fisiche effettivamente utilizzate per memorizzare i dati su calcolatore.
I modelli logici usati per le basi di dati sono:

- modello gerarchico
- modello reticolare
- modello relazionale
- modello ad oggetti.

In base al modello logico che seguono, le basi di dati si possono quindi suddividere nelle rispettive categorie (basi di dati gerarchiche, reticolari, relazionali, ad oggetti) e, conseguentemente, i DBMS funzionano sulla base del particolare modello logico.

Il modello relazionale è attualmente il modello logico più diffuso per organizzare i dati di una base di dati. Esso utilizza il costrutto di relazione, inteso in questo caso come concetto matematico proveniente dalla teoria degli insiemi (e non come legame/relationship), e organizza i dati secondo relazioni. Vedremo che una relazione potrà essere interpretata come una tabella a struttura fissa e vedremo che in sostanza una base di dati relazionale organizza (logicamente) i dati in tabelle.

A livello fisico, si modellizzano i dati per descrivere la loro organizzazione fisica sulla memoria del calcolatore.

Il modello gerarchico

Nel modello gerarchico i dati sono organizzati in record connessi tra loro secondo strutture ad albero (e quindi gerarchie, da cui il nome).
Ogni record del database, che non sia la radice dell' albero, deve avere uno e un solo padre. Possono quindi esserci due record, su

alberi diversi, che rappresentano la stessa informazione. Questo comporta problemi di ridondanza nel database e richiede controlli di consistenza durante il suo uso. Inoltre non è possibile memorizzare informazioni che non hanno padre.
La struttura gerarchica impone delle regole rigide sul modo di eseguire gli aggiornamenti e le interrogazioni: il livello più alto è il punto di accesso a tutte le informazioni. La cancellazione di un record del database comporta l'eliminazione di tutti i record dipendenti gerarchicamente da esso. L'aggiornamento di un dato richiede l'accesso e la modifica di diversi record per assicurare la consistenza del database.
Il modello gerarchico comporta la dipendenza dei programmi dalle strutture, quindi non possiamo modificare le strutture senza modificare i programmi.

Il modello reticolare

Nel modello reticolare i record sono legati tra loro con strutture ad anello (puntatori) che permettono all'utente di accedere ai dati più facilmente, senza i vincoli rigidi della struttura gerarchica. Ogni nodo può essere il punto di partenza per raggiungere un determinato campo.
Un record può avere uno o più record padre e ciò permette di evitare i problemi di ridondanza.
Rimangono il problema della dipendenza dei programmi dalle strutture e il problema della complessità delle strutture stesse che crescono in proporzione alla crescita dei dati.
Per modificare, anche parzialmente, le strutture bisogna chiudere il DB e ricrearlo.
Le strutture di dati utilizzate nel modello reticolare sono due, il record, con caratteristiche simili al concetto di relazione (o, più precisamente, al concetto di file di record in un linguaggio di programmazione, quale il Pascal o il Cobol), e il set, che permette di correlare record, per mezzo di catene di puntatori.
Una base di dati reticolare è definita con riferimento ad uno schema, che contiene tipi record collegati fra loro da tipi set.
Il modello reticolare è così chiamato poiché ogni suo schema può essere espressivamente rappresentato per mezzo di un grafo (o una rete, dall’inglese network), con i tipi record come nodi e i tipi set come archi.
Un tipo record è un tipo di dato composto, al quale è associato un nome. Esso è costituito da un insieme di campi. Ogni campo ha un nome che lo identifica all’interno del tipo record e ad esso è associato un tipo semplice, detto dominio del campo.
Un’occorrenza di un tipo record è una funzione che associa a ciascun campo un valore del corrispondente dominio.

È evidente la somiglianza fra il concetto di tipo record e quello di schema di relazione del modello relazionale (o di tipo record in un qualsiasi linguaggio di programmazione) e fra il concetto di occorrenza di record e quello di tupla relazionale (e di record).
È possibile specificare una chiave per ciascun tipo record, ma possono esistere tipi record senza chiave (accessibili attraverso catene di puntatori o scansioni sequenziali basate su un ordinamento fisico esistente fra i record).
In altri termini, possono quindi esistere occorrenze diverse di uno stesso tipo record con esattamente gli stessi valori.
Quindi a differenza di quanto accade per il modello relazionale (o meglio, per la sua formulazione astratta), per ciascun tipo record la base di dati contiene un multinsieme di occorrenze, non necessariamente un insieme.

Il modello relazionale

Nel modello relazionale i dati sono organizzati in tabelle che rappresentano sia le entità, sia le relazioni tra di esse: esistono quindi tabelle di entità e tabelle di relazioni.
Nel modello relazionale, a differenza dei precedenti, non c'è alcun meccanismo esplicito per rappresentare i legami logici tra i diversi tipi di record che non sia la relazione.
La modifica di un dato o di un legame comporta la manipolazione di un solo record di una tabella.
Nel modello relazionale, a differenza dei precedenti, si realizza l'indipendenza logica, è cioè possibile modificare le strutture senza dover modificare i programmi.
Si possono inoltre modificare le strutture a DB aperto, con gli utenti collegati.

Il modello relazionale si basa su due concetti, relazione e tabella, di natura diversa ma facilmente riconducibili l'uno all'altro.
Come accennato in precedenza, la nozione di relazione proviene dalla matematica, in particolare dalla teoria degli insiemi, mentre il concetto di tabella è semplice ed intuitivo.
La presenza contemporanea di questi due concetti, l'uno formale e l'altro intuitivo, è responsabile del grande successo ottenuto dal modello relazionale.
Infatti, le tabelle risultano naturali e comprensibili anche per gli utenti finali (che spesso le utilizzano in tanti contesti per scopi diversi, senza riferimento alle basi di dati).
D'altra parte, la disponibilità di una formalizzazione al tempo stesso semplice e chiara ha permesso anche uno sviluppo teorico a supporto del modello con risultati di interesse concreto.
Il modello relazionale risponde al requisito dell'indipendenza dei dati, che prevede una distinzione nella descrizione dei dati, fra il livello

fisico e il livello logico: gli utenti che accedono ai dati e i programmatori che sviluppano le applicazioni fanno riferimento solo al livello logico; i dati descritti al livello logico sono poi realizzati per mezzo di opportune strutture fisiche, ma per accedere ai dati non è necessario conoscere le strutture fisiche stesse.
Il modello relazionale fu proposto alla fine degli anni sessanta per permettere, a livello logico, una descrizione efficace.
Infatti, i modelli reticolare e quello gerarchico includevano espliciti riferimenti alla sottostante struttura realizzativa, attraverso l'uso di puntatori e l'ordinamento fisico dei dati.

È interessante notare che tutti i DBMS relazionali gestiscono il proprio dizionario dei dati (ovvero la descrizione delle tabelle presenti nella base di dati) mediante una struttura relazionale, cioè tramite tabelle.
La base di dati contiene quindi due tipi di tabelle: quelle che contengono i dati e quelle che contengono i metadati.
Questo secondo insieme di tabelle costituisce il catalogo della base di dati. Quasi sempre un sistema di gestione di basi di dati gestisce il catalogo mediante strutture analoghe a quelle che conservano l'istanza, per cui ad esempio una base di dati ad oggetti avrà un dizionario dei dati definito con un modello ad oggetti. In questo modo si fornisce all'amministratore di sistema un catalogo costruito con un modello conosciuto.
Si noti che nelle basi di dati esiste una parte sostanzialmente invariante nel tempo, detta "schema" della base di dati, costituita dalle caratteristiche dei dati, e una parte variabile nel tempo, detta "istanza" o stato della base di dati, costituita dai valori effettivi (dati contenuti.

Il modello reticolare e quello gerarchico hanno avuto origini diverse rispetto al modello relazionale, che è stato definito in astratto, con realizzazioni effettive arrivate molti anni dopo: il modello reticolare e quello gerarchico sono stati definiti come astrazione delle caratteristiche fondamentali di sistemi esistenti.
Di conseguenza, molte caratteristiche di questi due modelli ricordano da vicino tecniche di realizzazione fisica dei dati, piuttosto che caratteristiche intrinseche dei dati stessi, come avviene invece per il modello relazionale.
Un'altra importante differenza fra sistemi relazionali da una parte e sistemi reticolari e gerarchici dall'altra risiede nel fatto che nei primi le operazioni considerano globalmente insiemi di tuple, mentre nei secondi l'accesso viene effettuato considerando un record alla volta, e quindi la scansione di insiemi di record e delle loro connessioni con altri record, deve essere specificata esplicitamente dal programmatore per mezzo di cicli.

L’evoluzione del modello relazionale può considerarsi il modello a oggetti che estende alle basi dati il paradigma di programmazione ad oggetti.

3. PRINCIPALI CONCETTI E DEFINIZIONI

Un database relazionale è un insieme di tabelle che rappresentano ogni tipo di informazione.
Per reperire i dati è sufficiente conoscere la loro struttura logica e non è necessario conoscere la loro locazione fisica.
Una **Tabella** è un'insieme di informazioni attinenti tra loro. Essa rappresenta una entità o una relazione.
Una riga (o record o tupla) della tabella rappresenta una istanza di un'entità.
Una colonna della tabella rappresenta un attributo di quell'entità.
Nell'intersezione tra una riga e una colonna può esserci un solo valore, che può essere significativo o nullo (NULL VALUE, diverso da zero e da blank).
Non ci possono essere nomi di colonna duplicati.
L' ordine nel quale le righe sono contenute non ha importanza.
L' ordine nel quale le colonne sono contenute non ha importanza.

Affinché un RDBMS possa dirsi relazionale deve essere in grado di eseguire le tre operazioni relazionali di base: la proiezione, la selezione e il join.
La **Proiezione** è una visualizzazione "verticale" della tabella (solo alcune colonne).
La **Selezione** è una visualizzazione "orizzontale" della tabella (solo alcune righe che soddisfano una condizione).
Il **Join** è l'unione di record che sono memorizzati su tabelle diverse; permette di correlare dati contenuti in tabelle diverse, confrontando i valori contenuti in esse.

Il **Prodotto cartesiano** di due tabelle T1 e T2 è un insieme con tutte le possibili coppie di ogni record T1 con ogni record T2.
Il prodotto cartesiano completo non ha alcun interesse, occorre quindi selezionare da questo le righe significative, cioè quelle in cui il campo in comune tra le tabelle in join ha un contenuto uguale.
Queste sono le condizioni di join che legano insieme due o più tabelle in un **Join naturale**.
Il **Self join** è un join di una tabella con se stessa.
L' **Outer join** è un join tra due tabelle su una delle quali il record in join può mancare ma il record dell'altra tabella viene visualizzato comunque.

Tra le caratteristiche di un database relazionale vi è la possibilità di eseguire operazioni insiemistiche sulle righe estratte da due o più tabelle: unione, intersezione e differenza.
L'**Unione** consiste nell'estrazione di tutte le righe che compaiono in almeno una delle tabelle.
L'**Intersezione** consiste nell' estrazione delle righe comuni a tutte le tabelle.
La **Differenza** consiste nell'estrazione delle righe che compaiono nella prima tabella ma non nella seconda.

4. PROGETTO DELLE BASI DATI RELAZIONALI

La progettazione di una base di dati costituisce solo una delle componenti del processo di sviluppo, all'interno di un'organizzazione, di un sistema informativo complesso e va, quindi, inquadrata in un contesto più ampio, quello del ciclo di vita dei sistemi informativi, dalla fase di studio di fattibilità sino alla fase di implementazione.
Va detto che accanto alle attività citate, viene oggi spesso effettuata anche una attività detta di prototipizzazione, che consiste nell'uso di specifici strumenti software per la realizzazione rapida di una versione semplificata del sistema informativo, con la quale sperimentare le sue funzionalità. La verifica del prototipo può portare a una modifica dei requisiti ed una eventuale revisione del progetto.
Le basi dati costituiscono in effetti solo una delle componenti di un sistema informativo che tipicamente include anche i programmi applicativi, le interfacce con l'utente e altri programmi di servizio.
Comunque, il ruolo centrale che i dati hanno in un sistema informativo giustifica lo studio autonomo relativo alla progettazione delle basi di dati.
Questa maniera di procedere è peraltro coerente con l'approccio allo sviluppo dei sistemi informativi basato sui dati, in cui l'attenzione è centrata sui dati e sulle loro proprietà.
Con questo approccio viene prima progettata la base di dati e, successivamente, le applicazioni che la utilizzano.

Progettare un base di dati significa definirne struttura, caratteristiche e contenuto: si tratta, come è facile immaginare, di un processo nel quale bisogna prendere molte decisioni strategiche e l'uso di opportune metodologie è fondamentale per la realizzazione di un prodotto di alta qualità.
Una metodologia di progettazione consiste in:

- una decomposizione in passi successivi indipendenti dell'intera attività di progettazione
- una serie di strategie da seguire nei vari passi e alcuni criteri per la scelta in caso di alternative

- alcuni modelli di riferimento per descrivere i dati di ingresso e uscita delle varie fasi.

Le proprietà che una metodologia deve garantire sono principalmente:

- la generalità rispetto alle applicazioni e ai sistemi in gioco e quindi la possibilità di utilizzo indipendentemente dal problema allo studio e dagli strumenti a disposizione,
- la qualità del prodotto in termini di correttezza, completezza ed efficienza rispetto alle risorse impiegate,
- la facilità d'uso sia delle strategie che dei modelli di riferimento.

Nell'ambito delle basi di dati, si è consolidata negli anni una metodologia di progetto che ha dato prova di soddisfare pienamente le proprietà descritte.
Tale metodologia è articolata in tre fasi principali da effettuare in cascata e si fonda su un principio molto semplice ma efficace: quello di separare in maniera netta le decisioni relative a "cosa" rappresentare in una base di dati (prima fase), da quelle relative a "come" farlo (fasi successive).

Le tre fasi della metodologia di riferimento si distinguono in: la progettazione concettuale, la progettazione logica e la progettazione fisica.

La progettazione concettuale

Il suo scopo è quello di rappresentare le specifiche informali della realtà di interesse in termini di una descrizione formale e completa, ma indipendente dai criteri di rappresentazione utilizzati nei sistemi di gestione di basi di dati.
Il prodotto di questa fase viene chiamato schema concettuale e fa riferimento a un modello concettuale dei dati.
I modelli concettuali ci consentono di descrivere l'organizzazione dei dati ad un alto livello di astrazione, senza tener conto degli aspetti implementativi.
In questa fase infatti, il progettista deve cercare di rappresentare il contenuto informativo della base di dati, senza preoccuparsi né delle modalità con le quali queste informazioni verranno definite in un sistema reale, né dell'efficienza dei programmi che devono interagire con queste informazioni.

La progettazione logica

Consiste nella traduzione dello schema concettuale definito nella fase precedente, in termini delle strutture di rappresentazione proprie del tipo di sistema di gestione di base di dati a disposizione.
Il prodotto di questa fase viene denominato schema logico della base di dati e fa riferimento a un modello logico dei dati. Come noto, un modello logico ci consente di descrivere i dati secondo una

rappresentazione ancora indipendente da dettagli fisici, ma che può essere realizzata direttamente con un sistema di gestione di base di dati che adotta il modello logico scelto.
In questa fase, le scelte progettuali tengono conto anche di criteri di ottimizzazione delle rappresentazioni, in base alle operazioni da effettuare sui dati.
Si fa comunemente uso anche di tecniche formali di verifica della qualità dello schema logico ottenuto.
Nel caso del modello relazionale dei dati, la tecnica comunemente utilizzata è quella della normalizzazione.

La progettazione fisica

In questa fase lo schema logico viene completato con la specifica dei parametri fisici di memorizzazione dei dati.
Il prodotto di questa fase viene denominato schema fisico e fa riferimento a un modello fisico dei dati, che dipende dallo specifico sistema di gestione di basi di dati scelto e che tiene conto dei criteri di organizzazione fisica dei dati in quel sistema.

Vediamo in che maniera i requisiti della base di dati vengono utilizzati nelle varie fasi della progettazione.

Nella progettazione concettuale si fa uso soprattutto delle specifiche sui dati mentre le specifiche sulle operazioni servono solo a verificare che lo schema concettuale sia completo, contenga cioè le informazioni necessarie per eseguire tutte le operazioni previste.
Nella progettazione logica si fa invece riferimento allo schema concettuale per quanto riguarda i dati (cioè non si fa più uso diretto delle specifiche sui dati), mentre le specifiche sulle operazioni si utilizzano, insieme alle previsioni sul carico applicativo, per ottenere uno schema logico che renda tali operazioni eseguibili in maniera efficiente.
In questa fase bisogna conoscere il modello logico adottato ma non è ancora necessario conoscere il particolare DBMS scelto (solo la categoria cui appartiene). Infine, nella progettazione fisica si fa uso dello schema logico e delle specifiche sulle operazioni per ottimizzare le prestazioni del sistema.
In questa fase bisogna anche tener conto delle caratteristiche del particolare sistema di gestione di basi di dati utilizzato.
Il risultato della progettazione di una base di dati non è solo lo schema fisico, ma è costituito anche dallo schema concettuale e dallo schema logico.
Lo schema concettuale fornisce infatti una rappresentazione della base di dati di alto livello, che può essere molto utile a scopo documentativo, mentre lo schema logico fornisce una descrizione concreta del contenuto della base di dati che, prescindendo dagli aspetti implementativi, può essere utile come riferimento per le operazioni di interrogazione e aggiornamento.

Nel caso della progettazione di una base di dati relazionale il modello concettuale dei dati più diffuso è il modello Entità – Relazione.
A partire dai requisiti rappresentati da documenti e moduli di vario genere, acquisiti anche attraverso l'interazione con l'utente, viene costruito uno schema Entità-Relazione (rappresentato da un diagramma) che descrive a livello concettuale la base di dati.
Questa rappresentazione viene poi tradotta in uno schema relazionale, costituito da una collezione di tabelle.
Infine, i dati vengono descritti da un punto di vista fisico (tipo e dimensioni dei campi) e vengono specificate strutture ausiliarie per l'accesso efficiente ai dati.

Il modello Entità-Relazione si è affermato come standard di riferimento nelle metodologie di progetto di basi di dati e negli strumenti di ausilio alla progettazione di sistemi informativi.

Il modello Entità-Relazione

Il modello Entità-Relazione (E-R) è un modello concettuale di dati e fornisce quindi 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 negli elaboratori.
Questi costrutti vengono utilizzati per definire schemi che descrivono l'organizzazione e la struttura delle occorrenze dei dati, ovvero, dei valori assunti dai dati al variare del tempo.
I costrutti principali del modello sono: le entità, le relazioni e gli attributi.

Entità

Rappresentano classi di oggetti (fatti, persone) che hanno proprietà comuni ed "esistenza autonoma" ai fini dell'applicazione di interesse; ad es. città, dipartimento, impiegato, acquisto, vendita sono esempi di entità di un'applicazione aziendale.
Una occorrenza di una entità è un oggetto della classe che l'entità rappresenta.
Le città di Roma, Milano e Palermo sono esempi di occorrenze dell'entità Città, gli impiegati Marini e Ferrari sono invece esempi dell'entità impiegato.
Si osservi che una occorrenza di entità non è un valore che identifica un oggetto (per esempio, il cognome dell'impiegato o il suo codice fiscale) ma è l'oggetto stesso (l'impiegato in "carne e ossa").
Una interessante conseguenza di questo fatto è che una occorrenza di entità ha una esistenza (e una identità) indipendente dalle proprietà a esso associate (nel caso di un impiegato, il fatto di avere un nome, un cognome, una età, ecc.).
In questo il modello E-R presenta una marcata differenza rispetto al modello relazionale nel quale, per rappresentare un oggetto abbiamo invece bisogno di conoscere le sue proprietà (un impiegato viene

rappresentato da una ennupla contenente il nome, il cognome, l'età, ecc.).
In uno schema, ogni entità ha un nome che la identifica univocamente e viene rappresentata graficamente mediante un rettangolo con all'interno il nome dell'entità.

Relazioni (o associazioni)

Rappresentano legami logici, significativi per l'applicazione, tra due o più entità. Residenza è un esempio di relazione che può sussistere tra le entità Città e impiegato mentre esame è un esempio di relazione che può sussistere tra le entità studente e corso.
Una occorrenza di relazione è una ennupla (coppia nel caso di relazione binaria) costituita da occorrenze di entità, una per ciascuna delle entità coinvolte. La coppia di oggetti composta dall'impiegato Ferrari e dalla città di Maranello, è un esempio di occorrenza della relazione Residenza.
In uno schema E-R, ogni relazione ha un nome che la identifica univocamente e viene rappresentata graficamente mediante un rombo, con il nome della relazione all'interno, e da linee che connettono la relazione con ciascuna delle sue componenti; si osservi che possono esistere relazioni diverse che coinvolgono le stesse entità, come le relazioni residenza e sede di lavoro tra le entità impiegato e città.
Nella scelta dei nomi di relazione è preferibile utilizzare sostantivi invece che verbi, in maniera da non indurre a individuare un "verso" alla relazione.
È possibile avere relazioni che coinvolgono più di due entità. Ad esempio la relazione fornitura tra le entità fornitore, prodotto e Dipartimento descrive il fatto che ogni fornitore può fornire un certo prodotto a qualche dipartimento; un possibile insieme di occorrenza di questa relazione potrebbe stabilire che la ditta Pinto fornisce stampanti al dipartimento vendite e server al dipartimento sviluppo.

Attributi

Descrivono proprietà elementari di entità o relazioni di interesse ai fini dell'applicazione; per esempio data e voto sono possibili attributi per la relazione esame tra le entità studente e corso.
Un attributo associa a ciascuna occorrenza di entità (o di relazione) un valore appartenente a un insieme, detto dominio dell'attributo, che contiene i valori ammissibili.
Per esempio, l'attributo Cognome dell'entità impiegato può avere come dominio l'insieme delle stringhe di 20 caratteri, mentre l'attributo età può avere come dominio gli interi compresi tra 18 e 65.
Può risultare comodo, qualche volta, raggruppare attributi di una medesima entità o relazione che presentano affinità nel loro significato o uso: l'insieme di attributi che si ottiene in questa maniera viene detto attributo composto.

Possiamo, per esempio, raggruppare gli attributi via, numero civico e cap dell'entità persona per formare l'attributo composto Indirizzo.

Cardinalità delle relazioni

Viene specificata per ciascuna entità che partecipa a una relazione e descrive il numero minimo e massimo di occorrenze di relazione a cui le occorrenze delle entità coinvolte possono partecipare.
Indica quindi quante volte, in una relazione tra entità, un'occorrenza di una di queste entità, può essere legata a occorrenze delle altre entità coinvolte nella relazione. Per esempio se in una relazione Assegnamento tra le entità Impiegato e Incarico specifichiamo per la prima entità una cardinalità minima pari a uno e una cardinalità massima pari a cinque, vogliamo indicare che un impiegato può partecipare ad un minimo di una occorrenza e a un massimo di cinque occorrenze della relazione Assegnamento.
In altre parole, vogliamo dire che, nella nostra applicazione, ad un impiegato deve essere assegnato almeno un incarico, ma non più di cinque.
Se per l'entità Incarico specifichiamo una cardinalità minima pari a zero e una cardinalità massima pari a 50, imponiamo che un certo incarico può partecipare o a nessuna occorrenza oppure a 50 occorrenze al massimo della relazione Assegnamento.
Quindi, un certo incarico può essere assegnato a nessun impiegato oppure può essere assegnato a un numero di impiegati inferiore o uguale a 50.
Le cardinalità minima e massima delle entità di una relazione in uno schema E-R si specificano tra parentesi (1,5).

Cardinalità degli attributi

Può essere specificata per gli attributi di entità o relazioni e descrive il numero minimo e massimo di valori dell'attributo associati ad ogni occorrenza di entità.
Nella maggior parte dei casi, la cardinalità di un attributo è pari a (1,1) e viene omessa. Ad esempio una persona ha uno e un solo cognome, può avere o non avere la patente, ma se l' ha è unica, e può non avere automobili, ma può anche possederne più di una.

Identificatori delle entità (chiavi)

Vengono specificati per ciascuna entità di uno schema e descrivono i concetti (attributi e/o entità) dello schema che permettono di identificare in maniera univoca le occorrenze delle entità.
In molti casi, uno o più attributi di una entità sono sufficienti a identificare un identificatore: si parla in questo caso di identificatore interno (detto anche chiave).
Per esempio, un identificatore interno per l'entità Automobile con attributi Modello, Targa e Colore è l'attributo Targa, in quanto non

possono esistere due atutomobili con la stessa targa e quindi due occorrenze della entità Automobile, con gli stessi valori sull'attributo Targa.
Alcune volte però gli attributi di una entità non sono sufficienti a identificare univocamente le sue occorrenze.
Si possono avere ad esempio studenti iscritti a varie università ed inoltre due studenti iscritti a università diverse possono avere lo stesso numero di matricola.
In questo caso, per identificare univocamente uno studente, serve, oltre al numero di matricola, anche la relativa università.
L'identificatore interno è detto anche chiave.
Una *Chiave Primaria* (o *primary key)* è una colonna o un gruppo di colonne che identificano in maniera univoca (singola) ogni data riga rispetto alle altre. Tradotto in un linguaggio più semplice, è quell' insieme di informazioni che permette di distinguere ogni singolo record da ogni altro record.
Una *Chiave Esterna* (o *foreign key*) è invece una colonna presente in una tabella nella quale si registrano dati che sono la chiave primaria di un' altra tabella.

Abbiamo visto che gli schemi concettuali costituiscono utili strumenti nell'attività di progettazione di basi di dati.
In realtà, fornendo rappresentazioni dei dati coinvolti in una applicazione facili da comprendere, gli schemi E-R possono essere utilizzati indipendentemente dallo scopo finale di realizzare una applicazione.
Esistono diversi esempi di possibile uso degli schemi concettuali che prescindono dall'attività di progettazione:

- gli schemi E-R possono essere per esempio utilizzati a scopo documentativi, poiché sono facilmente comprensibili anche da non specialisti di basi di dati;
- possono essere utilizzati per descrivere e analizzare un sistema informativo già esistente e, nel caso di sistema costituito da diversi sottosistemi, c'è il vantaggio di poter rappresentare le varie componenti con un linguaggio astratto e quindi unificante;
- possono essere infine utilizzati per comprendere, in caso di modifica dei requisiti di una applicazione, su quali porzioni del sistema si deve operare e in cosa consistono le modifiche da effettuare.

La normalizzazione

Questo concetto permette di effettuare verifiche di qualità delle relazioni e costituisce quindi un utile strumento nell'ambito delle attività di progettazione di basi di dati.
Per gli schemi che non soddisfano la forma normale, è inoltre possibile definire un processo, detto di normalizzazione, che consente

di trasformare tali schemi in nuovi schemi che soddisfano la forma normale.
Il processo di normalizzazione è una realizzazione di passi per un'ottimizzazione delle tabelle in termini di ridondanza dei dati e dell'operatività.

5. INTEGRITÀ REFERENZIALE

I database relazionali rappresentano un'evoluzione del concetto di database. La loro architettura è basata sulla **relazione** che altro non è che una chiave di collegamento fra tabelle distinte. Nel caso dei database relazionali il motore di database verifica che i dati inseriti rispettino le cosiddette **regole d'integrità referenziale**. Ad esempio, se registriamo un ordine riferito ad un cliente inesistente, il motore di database si rifiuta di procedere avvertendoci che stiamo sbagliando l'operazione.
Se decidiamo di eliminare un cliente dal nostro archivio, il motore ci avverte che nel secondo schedario sono contenuti ordini relativi a quel cliente perciò sarà necessario mettere ordine al secondo schedario prima di cancellare il cliente.
Alcuni database relazionali sono così intelligenti che mettono ordine automaticamente: se cancelliamo un cliente, il motore di database provvederà a svuotare il secondo schedario eliminando tutti gli ordini del cliente depennato.

6. BASI DI DATI A OGGETTI

Le basi dati ad oggetti, sviluppate a partire dalla seconda metà degli anni ottanta, costituiscono una evoluzione delle basi di dati.
Esse rispondono a esigenze applicative recenti, emerse al di fuori delle tipiche applicazioni gestionali, per le quali il semplice modello relazionale dei dati si è dimostrato inadeguato.
I sistemi a oggetti integrano la tecnologia delle basi di dati con il paradigma a oggetti sviluppato nell'ambito dei linguaggi di programmazione e utilizzato, sul piano metodologico, nell'ambito dell'ingegneria del software.
Nelle basi di dati ad oggetti, ogni entità del mondo reale è rappresentata da un oggetto.
Esempi di oggetti possono essere:

- componenti elettronici progettati tramite un sistema di Computer Aided Design (CAD)
- specifiche e programmi da gestire in un ambiente di Computer Aided Software Engineering (CASE)

- componenti meccanici progettati tramite un sistema di Computer Aided Manufacturing (CAM)
- dati multimediali, che comprendono testi liberi, immagini e suoni
- dati spaziali o geografici, che descrivono ad esempio figure geometriche o mappe geografiche.

Queste applicazioni presentano caratteristiche abbastanza differenti fra loro, al punto che in alcuni di questi settori esistono DBMS specializzati in funzione dell'applicazione.
Però è comune a tutti questi campi applicativi una visione dei dati come oggetti complessi e unitari, male rappresentati tramite un ampio numero di tabelle, per i quali sono necessarie operazioni complesse di ritrovamento e modifica che sfruttano conoscenza specifica sugli oggetti, di assai difficile formulazione in linguaggi come SQL.

I database orientati ad oggetti (denominati OODB) vengono incontro a questa esigenza applicativa tramite sistemi (denominati OODBMS) che consentono:

- la specifica si strutture dati complesse con relazioni semantiche fra di dati (che somigliano alle strutture dati usate per la progettazione concettuale).
- la descrizione integrata dei dati e delle operazioni disponibili per il loro ritrovamento e modifica, nonché di operazioni che soddisfino specifiche esigenze applicative (ad esempio, operazioni per la rappresentazione tridimensionale di oggetti geometrici).
- una integrazione stretta con i linguaggi di programmazione ad oggetti, recuperando tipiche astrazioni di questi linguaggi (tra cui: incapsulamento, ereditarietà, overloading, overriding).

Nel modello dei dati ad oggetti, ogni oggetto ha un identificatore, uno stato e un comportamento; l'identificatore detto (OID) (Object Identifier), garantisce l'individuazione in modo univoco dell'oggetto nella base di dati, e consente di costruire riferimenti fra oggetti.
Ogni oggetto ha varie proprietà; lo stato di un oggetto è l'insieme dei valori assunti dalle sue proprietà in un determinato istante.
Infine, il comportamento di un oggetto è definito dai metodi (procedure, operazioni) che possono essere applicate all'oggetto stesso, normalmente predefiniti assieme all'oggetto.
Gli oggetti hanno un tipo e vengono raggruppati in classi.

I tipi sono astrazioni che consentono di descrivere sia lo stato che il comportamento.

Le classi descrivono sia la rappresentazione estensionale degli oggetti, sia la implementazione dei metodi relativi ad un tipo; il tipo

cioè descrive proprietà astratte, mentre la classe descrive la realizzazione di tali proprietà astratte tramite strutture dati e programmi.

La distinzione tra tipi e classi è uno degli argomenti più controversi nell'ambito dei linguaggi di programmazione e delle basi di dati ad oggetti; in particolare, sul piano teorico, esiste un terzo concetto, quello di estensione, che consente di inserire oggetti dello stesso tipo in collezioni differenti e dare nomi differenti a queste collezioni (ad esempio, il tipo cittadino potrebbe dar luogo a collezioni differenti, denominate ad esempio milanese e fiorentino).

7. BASI DI DATI E WORLD WIDE WEB

La rapida diffusione della rete Internet e la sua evoluzione costituiscono uno dei fenomeni più significativi di tutto il settore dell'informatica.
In quest'ambito, la tecnologia che più delle altre ha fatto registrare una crescita e un'accettazione oltre ogni aspettativa è quella del World Wide Web (www), che ha introdotto un nuovo paradigma di diffusione (per i fornitori) e acquisizione (per gli utilizzatori) delle informazioni, con facilità d'uso, flessibilità ed economicità indiscutibili, almeno con riferimento a molti contesti.
I protocolli e gli strumenti sviluppati ed utilizzati in Internet si sono diffuse anche per la realizzazione di reti private di organizzazioni ed aziende e viene allo scopo utilizzato il termine *Intranet.*
Poiché il Web ha come obiettivo la diffusione di informazioni e le basi di dati sono nate per organizzare e gestire le informazioni di interesse in un sistema informativo, possiamo esaminare la relazione che esiste tra le due tecnologie.

A prima vista, le differenze sono significative:

- il Web è nato soprattutto per gestire informazioni in primo luogo testuali e multimediali (immagini, audio, video): esso può essere definito come un sistema che gestisce documenti multimediali distribuiti.
- viceversa, le basi di dati sono state sviluppate soprattutto per gestire informazioni rappresentabili per mezzo di dati con struttura (relativamente) semplice.

Inoltre la tecnologia delle basi di dati sta cercando, per esempio attraverso i sistemi ad oggetti, di rispondere ad esigenze di applicazioni che richiedono la gestione di dati complessi (e multimediali); quindi possono esistere sistemi informativi in cui coesistono dati (e basi di dati) tradizionali e dati multimediali.
In secondo luogo, va notato che i browser (cioè gli strumenti che gli utenti usano per accedere al Web) costituiscono una sorta di

interfaccia generale per l'accesso a funzioni di vario tipo su sistemi locali o remoti: le informazioni di interesse possono essere documenti testuali o immagini, ma anche dati memorizzati in una base di dati.
Ad esempio il server Web di una azienda può mettere a disposizione il catalogo dei propri prodotti, che può essere memorizzato in una base di dati. In questo contesto esistono meccanismi per accedere ad una base di dati integrata con il web.

Bibliografia

Le dispense sono state tratte dal libro:
Atzeni, Ceri, Paraboschi, Torlone: Basi di dati: concetti, linguaggi e architetture. Ed. McGraw-Hill libri Italia Srl