

Sistemi informativi

Introduzione

Oggi l'Information Technology è impiegata in larga scala in tutte quelle attività che richiedono la gestione di dati e di informazioni. È impensabile al giorno d'oggi gestire e rendere competitiva un'azienda commerciale o una società di servizi senza l'ausilio di strumenti informatici (mentre nella tipologia di aziende citate l'uso degli strumenti informatici è indiscutibilmente un fattore critico di successo, nelle imprese industriali il discorso è più complesso, in quanto molteplici fattori concorrono in maniera significativa allo svolgimento del business aziendale. Grazie ai sistemi informatici è possibile, per esempio, fare acquisti con carta di credito anche a migliaia di chilometri di distanza dalla nostra banca, in Italia o all'estero. L'insieme di processi che si susseguono, da quando si inserisce la carta nell'apposito lettore a quando effettivamente è detratto il denaro dal conto corrente è chiamato "transazione". Ogni transazione è gestita da uno o più sistemi informatici che comunicano tra loro. Una qualsiasi attività, piccola o grande, individuale o cooperativa deve essere in grado di gestire in modo efficiente dati e informazioni legate all'attività stessa. Per far questo nel modo migliore, è necessaria la presenza di un sistema informativo.

Un sistema informativo non è legato esclusivamente alle tecnologie informatiche; infatti, è possibile progettarlo senza prevedere l'uso di sistemi hardware e software; esso è un sistema per organizzare le informazioni in modo che queste possano essere fruibili in maniera rapida ed efficiente.

Tornando indietro nel tempo, cerchiamo di immaginare il modo in cui poteva essere organizzato l'ufficio anagrafe comunale qualche decennio fa: alla richiesta di un certificato di nascita, l'impiegato allo sportello annotava su un foglietto di carta nome, cognome e data di nascita dell'utente. Quelle informazioni erano sufficienti per recarsi nella sezione dell'archivio giusta e prelevare il fascicolo riservato alle persone nate in quel giorno. In base al cognome era quindi possibile individuare l'atto di nascita desiderato e ottenere da esso tutte le informazioni per compilare il certificato di nascita. Dalla descrizione fornita non si può certo negare che l'ufficio fosse dotato di un sistema informativo: l'addetto, grazie alla preventiva organizzazione delle informazioni, sapeva sempre dove andare, in modo da soddisfare in un tempo più o meno lungo, le richieste dell'utente. Oggi, invece, all'addetto è sufficiente digitare nome, cognome e data di nascita su un terminale per ottenere in tempo brevissimo il certificato richiesto. Questo avviene perché il sistema informativo è automatizzato ed in tal caso si parla di sistema informatico.

La distinzione tra sistema informativo e sistema informatico esiste solo in teoria, in pratica non è più così netta come nel recente passato. Il termine sistema informativo al giorno d'oggi è solitamente legato alle tecnologie informatiche.

Dati e informazioni

Nell'ambito dei sistemi informativi esiste una sostanziale differenza fra dati e informazioni. Le informazioni sono rappresentate tramite dati, vale a dire un dato per diventare informazione deve essere associato ad un'interpretazione. Chiariamo il concetto con un esempio. Si ipotizzi di leggere i seguenti termini riportati su un foglio di carta:

- dott.
- Carlo
- Bianchi
- 780

Trattasi, nell'ottica ora esposta, di "dati": da essi non si ottiene infatti alcuna informazione; sapendo invece che Carlo Bianchi è il capo del personale della nostra azienda e 780 è il suo numero telefonico interno, allora tali dati assumono il ruolo di "informazione". Va notato come la condizione sufficiente affinché un dato diventi informazione è che esso sia interpretabile, non è invece necessario che sia qualcosa di corretto e sensato.

Ritornando ai sistemi informativi, i valori memorizzati in specifici byte di un file rappresentano semplicemente dei dati. Il sistema assocerà a tali dati un contesto in modo che possano diventare informazione. Per esempio, un programma potrebbe leggere i byte e interpretarli nel seguente modo: i byte da 0 a 19 rappresentano il cognome di un impiegato, da 20 a 39 il nome, da 40 a 47 la data di nascita, e così via.

Se poi i dati saranno utilizzati dal sistema assieme al contesto, allora l'utente riceverà l'informazione.

I sistemi informativi organizzano e rendono disponibili informazioni; per far questo accedono ai dati.

Database e DBMS

L'ufficio anagrafe non automatizzato da qualche decennio fa prevedeva esclusivamente supporti cartacei per memorizzare i dati. Questi erano organizzati in cartelle disposte in particolari settori dell'archivio. Inserire nuovi documenti, rispondere alle richieste degli utenti, riorganizzare particolari settori o storicizzare dati troppo vecchi erano mansioni tipiche del personale. Con l'avvento delle tecnologie informatiche, anche se le cose sono cambiate radicalmente, rimangono tuttavia interessanti analogie. L'insieme dei dati, denominato base di dati o database, è ora memorizzato permanentemente su memorie di massa (hard disk, cd-rom, nastri etc.). Tutto ciò che riguarda la gestione dei dati è delegato ai sistemi informativi e non più all'operatore di turno.

Alcune applicazioni software accedono ai dati facendo uso del file system, un modulo presente in tutti i sistemi operativi che si occupa della gestione dei file. In sistemi basati su questa metodologia, il database consiste in un insieme di file "proprietary" che l'applicazione gestisce mediante funzioni del sistema operativo.

Lo svantaggio di un simile approccio sta nel fatto che esiste una forte dipendenza tra dati e applicazione. Le applicazioni dipendono dal formato "fisico" dei dati, cioè dal momento in cui questi sono memorizzati all'interno dei file. Sarebbe sufficiente una piccola variazione in tale formato per rendere necessaria la riprogrammazione dell'applicazione, o di parte di essa. Evidentemente ciò rappresenta una notevole limitazione. Per questo motivo, la maggior parte dei sistemi informativi non è progettata in modo da accedere direttamente al file system, bensì a uno strumento software intermedio chiamato Database Management System (DBMS). A grandi linee, un DBMS può essere visto come uno strato di comunicazione tra applicazioni e dati. Utilizzando un DBMS le applicazioni non

accedono direttamente ai dati e, di conseguenza, non dipenderanno più dalla loro struttura. L'accesso al DBMS avviene in modo "logico" totalmente indipendente dalla rappresentazione fisica dei dati. Sarà compito del DBMS mappare le strutture logiche in rappresentazioni fisiche. Questo significa che il DBMS dovrà interpretare le richieste dell'applicazione, prelevare i dati, metterli in un formato standard e inviarli all'applicazione. Quest'ultima memorizzerà i dati in strutture interne e le utilizzerà, senza curarsi di sapere in che modo i dati siano memorizzati. Se nuove versioni del DBMS gestiranno il formato fisico dei dati in modo diverso, le applicazioni precedentemente progettate non necessiteranno di alcuna modifica. L'indipendenza applicazioni - dati non è però l'unico motivo per il quale i DBMS sono così diffusi. Affidarsi ad essi presenta altri importanti vantaggi riguardanti aspetti come protezione di accesso ai dati, riduzione di ridondanze e inconsistenze, sicurezza e ottimizzazione nell'uso dei dati.

Lo scambio di informazioni tra applicazione e DBMS avviene attraverso speciali linguaggi di interrogazione noti ad entrambi. Le applicazioni inviano comandi che rappresentano richieste di vario genere; il DBMS interpreta tali comandi in modo da soddisfare le richieste dell'applicazione. Una richiesta (o più comunemente query) può anche essere inoltrata semplicemente da un utente, attraverso speciali tool messi a disposizione dal DBMS, senza bisogno di programmare un'applicazione ad hoc.

Esistono varie tipologie di DBMS che si differenziano tra loro in base al meccanismo di organizzazione logica dei dati: quelli analizzati in questa sede sono chiamati DBMS relazionali ed usano il linguaggio di interrogazione denominato SQL; altri modelli, adottati per lo più in passato, sono il modello reticolare e quello gerarchico.

Ciclo di vita di un sistema informativo

Da quanto abbiamo potuto apprendere un sistema informativo può essere considerato una combinazione di software, hardware e dati. Come tutto ciò che ruota attorno all'Information Technology, anche i sistemi informativi hanno un ciclo di vita: nascono, vivono per un po' di tempo e poi muoiono.

L'esigenza di un particolare prodotto software che gestisca più aspetti di una realtà aziendale porta alla prima fase di sviluppo di un sistema informativo, ovvero lo studio di fattibilità.

Mediante tale studio, un team di esperti composto da personale interno alla società e da consulenti esterni valuta costi e benefici del nuovo sistema e decreta se il progetto può andare avanti o meno. Da questo momento in poi si entra nella fase di acquisizione dei requisiti che il sistema informativo deve soddisfare. Analisti e utenti saranno coinvolti nella raccolta di tutte le informazioni rilevanti da utilizzare in fase di progettazione vera e propria del sistema.

Questa seconda fase porta come risultato a un insieme di documenti e altre risorse che stabiliscono le funzionalità che il sistema dovrà possedere.

La fase successiva è quella relativa al modello dei dati. L'obiettivo di questa fase è di produrre una rappresentazione formale delle informazioni contenute dal sistema. Per passare alla fase successiva, il modello dati proposto deve essere approvato dagli utenti e dagli sviluppatori.

Una volta approvato, il modello dei dati è pronto per essere tradotto in un modello logico e infine in un database. A questa fase lavorano analisti, database designer e sviluppatori esperti.

Una volta che la base di dati è stata disegnata, le fasi che si susseguiranno sono: sviluppo dell'applicazione che accede ai dati, documentazione e test. Lo sviluppo dell'applicazione è delegato a un team di software designer e di programmatori. In base ai documenti prodotti dagli analisti e alle specifiche della base dei dati, i programmatori possono sviluppare le applicazioni che, attraverso il DBMS, accedono ai dati. Questi inoltre documenteranno le tecnologie software utilizzate, mentre utenti chiave documenteranno le funzionalità del sistema informativo. Un team di tester, solitamente composto da programmatori e utenti, verificherà il corretto funzionamento di ogni singola funzionalità del sistema.

Solo dopo aver superato tutte queste fasi, il sistema informativo entrerà in fase di esercizio; durante questa fase, che può essere più o meno lunga, il sistema potrebbe subire variazioni dovute al cambiamento di esigenze aziendali o alla scoperta di anomalie. Ognuna di queste variazioni potrebbe portare a rivedere una delle fasi precedentemente discusse. Nel momento in cui un nuovo sistema informativo entrerà in esercizio, sostituendone uno vecchio, quest'ultimo morirà. Questa è l'ultima fase.

Tipologie di sistemi informativi

Con il termine "architettura client-server" si intende una ben definita tecnologia che può essere considerata il punto di arrivo nell'evoluzione dei sistemi informativi. In questa sezione esamineremo alcune delle diverse architetture che sono emerse nel corso degli anni; alla fine dovremo essere in grado di valutare se effettivamente un sistema informativo è stato progettato secondo i canoni client-server o meno.

Architettura centralizzata

La prima architettura apparsa sullo scenario dell'elaborazione dati è quella basata su sistemi host; nell'architettura centralizzata l'elaborazione di tutti i processi è affidata all'host computer, potente e costosissima macchina che, nelle prime versioni, poteva occupare anche stanze intere. In questo tipo di sistemi, la base dati e le applicazioni risiedono sull'host, al quale gli utenti possono accedere mediante terminali "stupidi". La definizione "stupido" risulta molto appropriata poiché tali terminali, composti solitamente da un monitor, una tastiera e a volte da un mouse, non hanno nessuna possibilità di eseguire processi. Essi rappresentano lo strumento con cui l'utente invia e legge le informazioni. L'host invia ai terminali la codifica binaria dei caratteri che il terminale deve visualizzare. Viceversa il terminale cattura le azioni dell'utente, solitamente informazioni sui caratteri della tastiera digitati oppure sulle coordinate della zona in cui è avvenuto un clic del mouse e le spedisce al host.

Ancora oggi l'architettura centralizzata è la soluzione adottata da numerosi sistemi informativi, specialmente in ambito bancario, di gestione anagrafica della popolazione, per prenotazioni aeree su scala mondiale e in tutti i contesti dove esiste una forte specializzazione e un'esigenza di altissime prestazioni. Infatti, un sistema informativo basato sull'architettura centralizzata è fortemente specializzato in un determinato settore, di conseguenza è poco flessibile. Questa caratteristica rende i sistemi centralizzati poco adatti per quei contesti caratterizzati da rapidi mutamenti sia tecnologici sia relativi alle necessità dell'utenza finale.

Inoltre i costi di un sistema informativo basato su host sono molto elevati, il rapporto prezzo/prestazioni con altri sistemi di calcolo è nettamente sfavorevole.

Architettura basata su minicomputer

Alla fine degli anni Settanta comparvero e si diffusero i primi minicomputer. Di conseguenza, i produttori di sistemi informativi iniziarono a sviluppare i primi sistemi nei quali l'elemento centrale era il minicomputer. Un sistema basato su minicomputer differisce da uno basato sull'architettura centralizzata per una sola caratteristica: il minicomputer ha sostituito quindi l'host. Tecnicamente parlando non è cambiato nulla o quasi: siamo sempre in presenza di una macchina che ha alle spalle tutta l'elaborazione e di tanti terminali "stupidi". Nonostante questo, per le piccole e medie aziende l'introduzione del minicomputer è stata salutata come una vera e propria manna dal cielo: infatti, il costo di un minicomputer è notevolmente inferiore a quello di un host, conseguentemente molte aziende sono state in grado di sostenere le spese per l'introduzione di un sistema informativo.

Architettura basata su LAN

I primi anni Ottanta hanno visto il diffondersi del personal computer, computer dai costi ridotti, di scarse prestazioni, nati per essere usati in piccolissime attività ma soprattutto a casa e prevalentemente per giocare. Verso la metà degli anni Ottanta, quando i PC avevano raggiunto già discrete prestazioni e un costo relativamente basso, iniziò il diffondersi delle reti locali (LAN). L'architettura basata su LAN è l'antitesi dell'architettura centralizzata. Infatti, tutta l'elaborazione dei processi risiede su PC: grazie alla rete locale è possibile condividere una grande varietà di risorse, come per esempio database, file, stampanti, unità disco etc. In un sistema informativo basato su LAN esistono n postazioni "intelligenti" che accedono a un server passivo la cui unica funzionalità è quella di essere un contenitore di dati e risorse.

Nel momento in cui uno dei PC accede a un database, ciò che viaggia effettivamente sulla LAN sono proprio i byte che compongono i settori di disco sul quale i dati sono memorizzati.

Uno dei vantaggi indiscussi dell'architettura LAN è la possibilità di potenziare i PC mediante risorse condivise; inoltre tali architetture sono flessibili, affidabili, facilmente espandibili e riconfigurabili. Lo svantaggio numero uno è rappresentato dalle scarse prestazioni. Un'applicazione che gira su PC e accede a un database condiviso deve effettuare tanti accessi al disco che ospita il database per ogni record che desidera leggere: quando il numero di record è molto alto, i tempi di attesa potrebbero diventare insostenibili.

Da quanto detto, quindi, si sente l'esigenza di un'architettura che sia una via di mezzo tra quella centralizzata e quella basata su LAN, che possibilmente sia flessibile, relativamente economica e ad alte prestazioni.

Architettura client-server

L'architettura client-server rende possibile una divisione equa dei compiti tra nodi client e nodi server. L'insieme delle applicazioni client, detto anche front-end, rappresenta l'interfaccia utente del sistema informativo. In essa è contenuta la logica per le richieste da inoltrare al server e la gestione dei risultati. Il server è inoltre composto da quell'insieme di processi, denominato back-end, che elabora le richieste del client e fornisce i risultati.

Nelle architetture client-server il back-end del sistema è quasi sempre rappresentato dal DBMS. Nell'architettura client-server basata su database relazionali, le informazioni che viaggiano sulla rete dal client verso il server sono richieste formulate in SQL. Viceversa, il server invia al client dei set di risposta, contenenti i dati richiesti, composti da una sequenza di righe prelevate dalle tabelle del database.

Molti sono gli aspetti che hanno sancito il successo di questo particolare tipo di architettura. Dal punto di vista della progettazione i sistemi client-server sono costituiti da moduli finalizzati a uno scopo ben definito. Il front-end è nettamente separato dal back-end in termini di progettazione, piattaforma e tecnologia hardware. I sistemi client-server sono una valida alternativa ai sistemi proprietari dei grandi produttori e si configurano come sistemi aperti. Gli utenti di un sistema client-server hanno a disposizione risorse individuali e centralizzate unificate da un'architettura unitaria di sistema informativo.

a. le applicazioni client

Le applicazioni client hanno, come prima responsabilità, il compito di collegarsi al server. Un client chiederà login e password oppure le preleverà da qualche parte (per esempio nel registro di configurazione di Windows). È anche compito del client terminare la connessione quando questa non è più necessaria e gestire eventuali errori dovuti a malfunzionamenti della rete, del server o del client stesso.

Le applicazioni client forniscono delle maschere progettate ad hoc per facilitare l'utente nel formulare richieste verso il server. Rientra anche nei compiti del client controllare la validità delle informazioni che l'utente inserisce. Per esempio, se in una maschera è richiesto di inserire il codice cliente, e questo è memorizzato nel DBMS come valore numerico, allora l'applicazione dovrebbe controllare se è stato effettivamente introdotto un numero. In caso contrario deve segnalare l'errore e invitare l'utente a digitare un numero valido. È compito del client gestire i risultati forniti dal server in risposta a particolari richieste. Questi possono essere stampati su video, su stampante. Utilizzati per completare delle maschere etc. Infine, il client ha il dovere di gestire gli errori che possono essere dovuti a problemi di connessione oppure a operazioni errate sui dati. L'applicazione dovrebbe fare tutto il possibile per correggere l'errore o mettere l'utente in condizione di capire dove questo si sia verificato. Solo in casi estremi, quando non si riesce in nessun modo a riscontrare l'effettiva natura dell'errore, dovrebbe apparire un messaggio d'errore fatale.

b. i compiti del server

Nell'architettura client-server il server non è un depositario passivo di dati ma piuttosto un componente intelligente dal quale dipendono la corretta organizzazione e gestione dei dati. Il server controlla la validità e l'integrità dei dati ogni volta che applicazioni client autorizzate accedono al database. Questo significa: aggiornare correttamente le tabelle, gli indici e gli oggetti temporanei, ripristinare transazioni non terminate correttamente etc.

Il server compila ed esegue query, esegue stored procedure e restituisce i risultati al client chiamante. Inoltre è compito del server gestire le autorizzazioni d'accesso, quindi la sicurezza della base di dati e correggere errori interni, che nella maggior parte dei casi i client non vedranno mai. La maggior parte di queste operazioni sono effettuate in maniera trasparente dai vari DBMS. È in ogni caso compito dei progettisti software e degli amministratori del database utilizzare e configurare al meglio il DBMS per ottenere affidabilità, alte prestazioni e una corretta gestione della sicurezza.

Modelli dei dati e database relazionali

Potrebbe essere lecito porsi la domanda su che cosa spinga una persona a decidere di progettare un database. La risposta va cercata in un'esigenza, sua o di un altro soggetto, di rendere automatiche alcune operazioni inerenti la propria attività. Progettare una base di dati è un'operazione che scaturisce da esigenze reali, da entità concrete o astratte presenti nella vita quotidiana. Se il consiglio di amministrazione di un'azienda decide di commissionare la realizzazione di un sistema informativo, i progettisti del database devono scontrarsi con aspetti della realtà relativi al ciclo produttivo dell'azienda. Il compito dello staff di progettazione è di modellare la realtà che c'è intorno, individuare gli oggetti utili ai fini del sistema e, da questi, organizzare i dati e le informazioni in modo preciso ed efficiente. Questa preliminare fase di analisi è necessaria per modellare la "realtà di interesse", termine tecnico che sta ad indicare tutto ciò che di reale ruota attorno ad un'attività sia essa di carattere gestionale, amministrativo, scientifico o altro. La realtà di interesse è l'insieme di attività e processi aziendali, al quale il team di analisti e progettisti è interessato.

Schemi e modelli

Uno degli obiettivi principali riguardante l'analisi della realtà d'interesse è di pervenire alla struttura (o schema) finale del database. Questo processo è composto da tre passi principali: i primi due portano alla realizzazione di altrettanti schemi intermedi, il terzo ha come risultato lo schema finale. Quindi, partendo dalla realtà di interesse, è possibile ottenere lo schema fisico, ovvero la struttura finale del database, attraverso le seguenti elaborazioni:

- realtà di interesse → schema concettuale
- schema concettuale → schema logico
- schema logico → schema fisico

Uno schema è una rappresentazione formale della realtà; esso può essere visto come un insieme di simboli posti in corrispondenza con elementi della realtà. I processi che consentono il passaggio tra uno schema e il successivo sono realizzati attraverso modelli. Un modello è un insieme di regole e di convenzioni che portano alla realizzazione di uno schema. La realtà di interesse può essere modellata in uno schema concettuale attraverso un modello concettuale: così come uno schema concettuale può essere trasformato in uno schema logico mediante un modello logico, uno schema logico può essere trasformato in uno schema fisico grazie ad un modello fisico. Il modello è quindi un insieme di regole che portano alla definizione di uno schema.

Modelli concettuali

La progettazione dello schema concettuale di un database è senza dubbio una delle operazioni più delicate nella progettazione di un sistema informativo. Una cattiva progettazione influenzerà le funzionalità e l'efficienza dell'intero sistema. Il punto cruciale si concentra sul fatto che la realtà di interesse è sensibile alle diverse interpretazioni; può essere considerata una porzione di mondo così come è percepita da chi è incaricato di realizzare il modello. Sta di fatto che non sempre questi percepisce bene e le conseguenze influenzeranno negativamente tutte le attività successive.

I modelli di dati usati nella progettazione concettuale sono definiti anche modelli concettuali o semantici. È molto importante tenere presente che la realizzazione di uno schema concettuale è totalmente indipendente da qualsiasi piattaforma hardware o software e da qualsiasi DBMS. Corollario: gli schemi concettuali potrebbero esistere anche senza i computer.

Il modello semantico più in voga è quello sperimentato da Peter P. Chen nel 1976 chiamato entità-relationship model (modello entità-relazione). Attraverso questo modello, dalla realtà sono estrapolati oggetti denominati entità, aventi caratteristiche proprie rappresentate mediante attributi.

Per esempio, riferendoci ad un'azienda per il noleggio di automobili, alcune delle entità estrapolate dalla realtà potrebbero essere le seguenti: *cliente*, *auto*, *contratto*, *pagamento*...

Ciascuna entità sarà caratterizzata da un insieme di attributi. Per esempio, gli attributi dell'entità *Cliente* potrebbero essere: codice, cognome, nome, data di nascita, indirizzo, numero patente, tipo patente, codice fiscale.

Individuare e distinguere gli attributi dalle entità è uno degli aspetti più delicati dell'analisi concettuale. Il legame tra un'entità e i suoi attributi non è assoluto ma varia in funzione del problema che si sta affrontando. In un sistema informativo per la gestione di un'agenzia turistica l'entità *Cliente* potrebbe avere i seguenti attributi: codice, cognome, nome, data di nascita, indirizzo, numero di passaporto, nazionalità, codice fiscale.

Come si può notare, l'entità *Cliente* per il sistema di autonoleggio e quella per l'agenzia turistica sono simili ma non uguali. Il contesto in cui operano è diverso; di conseguenza, nel primo caso è stato necessario introdurre attributi come "numero patente" e "tipo patente", che nel secondo caso non sono necessari; viceversa, l'entità *Cliente* relativa all'agenzia turistica presenta gli attributi "numero passaporto" e "nazionalità" che nel caso dell'autonoleggio non sono rilevanti.

In genere, le diverse entità individuate in un determinato contesto non sono indipendenti l'una dall'altra, bensì esistono differenti tipi di relazioni che legano un'entità all'altra. Ad esempio, l'entità *Cliente* relativa al sistema di autonoleggio avrà di certo qualche relazione con l'entità *Auto*: infatti un cliente è una persona che noleggia un'auto.

Modelli logici

Il modello logico consente la trasformazione di uno schema concettuale in uno logico. Lo schema logico è una collezione di strutture che rappresentano il database. Esso è totalmente indipendente dalla piattaforma hardware o dal sistema operativo, ma dipende fortemente dal tipo di DBMS sul quale dovrà funzionare. Questo accade perché generalmente i DBMS sono sviluppati per funzionare con un determinato modello logico. Di conseguenza, chi realizza lo schema deve avere bene in mente il DBMS target e soprattutto il modello logico che questo implementa. Tra i modelli logici il più diffuso è senza dubbio il modello relazionale, nel quale lo schema può essere visto, in prima approssimazione, come una collezione di tabelle e di relazioni tra tabelle. Tra i DBMS relazionali commerciali i più noti sono Oracle, SQL Server, Informix Sybase e DB2. Esistono anche versione shareware o freeware come MySQL:

un database relazionale economico, robusto ed efficiente. Per applicazioni di piccole dimensioni un ottimo prodotto potrebbe essere Microsoft Access, incluso nel diffuso pacchetto Office.

Il modello gerarchico e quello reticolare sono due esempi di modelli logici usati in passato, ma attualmente non molto diffusi. Nel primo, le informazioni sono strutturate secondo uno schema ad albero. Nel secondo, il database è modellato come un reticolo, una struttura dati che sostanzialmente differisce da un albero giacchè ogni nodo può avere più di un nodo padre.

Negli ultimi anni stanno prendendo piede database basati sul modello logico orientato agli oggetti: seguendo questo modello, i dati vengono strutturati in base al paradigma orientato agli oggetti. Lo schema sarà quindi composto da classi, attributi, associazioni ed aggregazioni con altre classi. Usando un linguaggio e un database orientati agli oggetti, la memorizzazione degli oggetti potrà avvenire senza dover prevedere il mapping tra questi e le tabelle del database relazionale.

Modelli fisici

Un modello fisico è l'insieme di regole che consente di trasformare uno schema logico in uno fisico. Uno schema fisico descrive il modo in cui il modello logico sarà memorizzato su una particolare piattaforma hardware/software. Generalmente, il DBMS trasforma un modello logico in uno fisico automaticamente e in modo totalmente trasparente. Nel caso in cui non sia previsto l'utilizzo di un DBMS il programmatore dovrà implementare a mano tutto il codice necessario per l'organizzazione e la memorizzazione dei dati. Per attuare quest'ultima operazione sarà necessario scomodare il file system di un particolare sistema operativo. Le regole di progettazione sono dettate dalle caratteristiche del sistema ospite, il livello d'astrazione in questa fase è perciò molto ridotto.

Il modello entità - relazione

Come già detto, il modello entità - relazione è il più diffuso tra i modelli concettuali. Questo modello è usato ormai da un quarto di secolo nella progettazione della base di dati di numerosi sistemi informativi attualmente in uso. Il suo successo è dovuto a una relativa semplicità d'utilizzo e, grazie ai diagrammi entità-relazione, a una chiara e intuitiva lettura. Inoltre la trasformazione da un diagramma entità - relazione a uno schema relazione è pressoché immediata. Questo ha decretato il definitivo successo del modello.

Entità

L'entità è uno dei due concetti basilari del modello entità - relazione (l'altro è ovviamente la relazione). Un'entità rappresenta un insieme di oggetti della realtà d'interesse. La prima attività relativa alla progettazione concettuale è proprio l'individuazione delle entità; le proprietà di cui godono le entità sono rappresentate mediante attributi; in un sistema di gestione ordini relativo a un'attività commerciale, alcune entità potrebbero essere: *Ordine*, *cliente*, *fattura*, *articolo*. L'entità *Ordine* potrebbe avere come attributi: numero, data, tipo.

I dati relativi ad un'entità sono chiamati elementi o occorrenze dell'entità. Alcune metodologie di progettazione e sviluppo, basate sul modello entità - relazione, prevedendo una classificazione delle entità. Un'entità si dice fondamentale se non dipende da nessun'altra entità del sistema. Per indipendenza si intende che un'entità, anche presa singolarmente, è significativa. In un sistema per la gestione degli ordini, l'entità *Ordine* è sicuramente fondamentale; così come lo saranno l'entità *Cliente* e l'entità *Articolo*.

Un secondo tipo di entità, chiamata caratteristica, è quella che assume significato solo se messa in corrispondenza con altre entità. In un sistema informativo per la gestione del personale, potrebbero essere presenti le entità *Impiegato* e *Familiare*. Questa ultima rappresenta i singoli componenti della famiglia dell'impiegato. È chiaro che l'entità *Familiare* non è fondamentale ma caratteristica. Infatti, essa assume significato solo se messa in corrispondenza con uno degli elementi dell'entità *Impiegato*. Nel momento in cui un impiegato lascia l'azienda dal database del sistema sarà cancellato non solo il corrispondente elemento dell'entità *Impiegato* ma anche tutte le entità *Familiare* ad esso collegate.

Relazione

Le entità possono essere legate tra loro attraverso relazioni. Ritornando all'entità *Ordine*, esisterà sicuramente una relazione con l'entità *Cliente* poiché un ordine sarà emesso nel momento in cui un cliente decide di acquistare qualcosa. Più precisamente, ad ogni ordine è associato un singolo cliente, mentre un cliente può aver richiesto zero o più ordini. Se consideriamo invece la relazione tra l'entità *Ordine* e l'entità *Articolo*, le cose cambiano; a un ordine possono essere associati uno o più articoli, un articolo può comparire in zero o più ordini. Ovviamente, la relazione che intercorre tra *Ordine* e *Cliente* è diversa da quella che lega *Ordine* e *Articolo*. Ciò che cambia è la cardinalità della relazione.

I prodotti finali dell'analisi dati concettuale basata sul modello entità-relazione sono sostanzialmente due. Il primo è un documento nel quale sono descritte le entità e gli attributi attraverso commenti e descrizioni.

Esistono tre tipi fondamentali di relazioni tra entità:

- 1:1 (uno a uno): due entità E e F sono in relazione 1:1 se a ogni elemento di E può corrispondere un solo elemento di F e viceversa. Un esempio di relazione 1:1 è quella che generalmente lega le nazioni alle proprie capitali. A ogni nazione è associata una sola capitale e viceversa. Graficamente una relazione è denotata con un arco che collega le entità e sul quale è specificato il tipo di relazione.
- 1:N (uno a molti): due entità E e F sono in relazione 1:N se a ogni elemento di E possono corrispondere più elementi di F, mentre a ogni elemento di F può corrispondere un solo elemento di E. La relazione tra l'entità *Ordine* e l'entità *Cliente* è di questo tipo: un cliente può effettuare più ordini, a un ordine è associato un singolo cliente.
- N:N (molti a molti): due entità E ed F sono in relazione N:N se a ogni elemento di E possono corrispondere più elementi di F e a ogni elemento di F possono corrispondere più elementi di E. Le entità *Ordine* e *Articolo* sono in relazione N:N. Un ordine può contenere più articoli, un articolo può essere contenuto in più ordini.