Concetti generali di Basi di Dati

Informazioni e dati

Significato di INFORMAZIONE

Significato di DATO

Quando un DATO diventa INFORMAZIONE

DATO qualitativo e DATO quantitativo

Sistemi informativi, Sistemi Informatici, Basi di dati

Importanza delle INFORMAZIONI nelle organizzazioni soprattutto moderne

Concetto di Sistema Informativo

Esempi di raccolte di INFORMAZIONI
- tavolette di argilla di UR
- biblioteca di Alessandria
- censimento della popolazione dell'Impero Romano
- mappe nautiche e portolani medioevali
- banche rinascimentali fiorentine

Cosa e' l' INFORMATICA

Cos'è un Sistema INFORMATICO

Esempi di sistemi INFORMATICI
- anagrafe comunale
- banca
- IATA (biglietti aerei)
- Amazoon
- eBay
- CERN

Centralità del DATO nei moderni sistemi INFORMATICI

Analisi “Dimensionale” delle moli di DATI
- Numerosità
- Precisione
- Variabilità
- Complessità
- Multidimensionalità
- Multimedialità

Caratteristiche e misure degli esempi di sistemi INFORMATICI
- anagrafe comunale
  - persistenza dei dati
- banca
  - atomicità delle operazioni
  - volumi nel tempo (fault tollerant)
- IATA (biglietti aerei)
  - coordinamento tra più parti (transazionalità)
  - condivisione dei dati
- Amazoon
  - accesso concorrente
  - semplicità
- eBay
  - adattabilità
  - riservatezza
- CERN
  - volumi infiniti !!!!

Necessità di una gestione dei DATI

** Sistemi di gestione di basi di dati (DBMS)

Definizione del termine DBMS

Come funzionava l'INFORMATICA prima dei DBMS
- Schede perforate, nastri, dischi
- Programmi ed archivi (esempio C)
- File con struttura standard e dichiarata (esempio COBOL)
- Condivisione di struttura tra più programmi (librerie COBOL)

Problemi a cui i DBMS cercano di dare una risposta
- grandi moli di dati
- efficienza (non riscrivere ogni volta gli stessi algoritmi più o meno ottimizzati)
- efficacia
- condivisione dei dati tra più applicazioni
- persistenza dei dati nel tempo
- affidabilità e meccanismi di sicurezza (backup, recovery)
- riservatezza dei dati (controllo accesso e permessi differenziati)

Tipologie di Modelli dei dati

Rappresentazione dell'informazione
- le scritture contabili
- i piani dei conti
- le schede anagrafiche
- le registrazioni delle osservazioni astronomiche

Il Modello dei dati
- Visione astratta e condivisibile delle INFORMAZIONI
  - aggiungere un significato ai DATI
- Schema dei dati
- Istanza

Modelli implementativi
- Modello a FILE
  - elenchi di scritture contabili da sfogliare avanti e indietro
- Modello Gerarchico
  - piano dei conti di un'azienda
- Modello Reticolare
  - le schede anagrafiche di un ufficio comunale
- Modello Relazionale
- Modello ad Oggetti

Concetti centrali del modello Relazionale
- Relazione
- Attributo
- Record o Tupla (Ennupla)

Livelli di astrazione ed indipendenza dei dati

Equivalenza espressiva di diversi modelli

I “banali dettagli implementativi”

Modi diversi di vedere la stessa struttura

Livelli di astrazione
- Livello fisico (o interno)
- Livello logico (aggiunta del significato al livello fisico, senza dettagli implementativi)
  - utilizzato dai programmi
  - indipendenti dalla struttura fisica
- Livello esterno (altri modi per vedere un unico livello logico)
- Livello concettuale
  - astrae dai dettagli del livello logico cercando di modellare la realtà

Esempio di livello fisico
- Struct in C
- DataDivision in COBOL
- Scheda per lettura ottica (tipo modulo INPS)

Esempio di livello logico
- Archivio anagrafico dell'ufficio del personale di un'azienda

Esempio di livello esterno

Organigramma di un'azienda partendo dall'archivio dell'ufficio del presonale

Esempio di livello concettuale
Modello delle entità e delle loro relazioni all'interno di un'azienda astratta

Indipendenza tra livelli
Indipendenza fisica
Indipendenza logica

Vantaggi e svantaggi di un DBMS

Vantaggi
unicità del dato
coerenza
condivisibilità
indipendenza del dato
disponibilità di strumenti ottimizzati

Svantaggi
necessità della presenza del DBMS
impossibilità di effettuare decisioni ed ottimizzazioni di microdettaglio
competenze
costi

Esempi e storia dei DBMS
ORACLE (leader)
IBM-DB2 (da non confondere con DBII e DBIII)
MS-SQLServer
Informix, Ingres, SyBase (sic transeat gloria mundi)
Paradox, Access, FileMaker (i piccoli)
AS400 (un mondo a parte)
MySQL e PostgreSQL (il nuovo mondo)

INTEGRAZIONE

Livelli di astrazione ed indipendenza dei dati

Equivalenza espressiva di diversi modelli

I “banali dettagli implementativi”

Modi diversi di vedere la stessa struttura

Livelli di astrazione
- Livello fisico (o interno)
- Livello logico (aggiunta del significato al livello fisico, senza dettagli implementativi)
  - utilizzato dai programmi
  - indipendenti dalla struttura fisica
- Livello esterno (altri modi per vedere un unico livello logico)
- Livello concettuale
  - astrae dai dettagli del livello logico cercando di modellare la realtà

Esempio di livello fisico
- Struct in C
- DataDivision in COBOL
- Scheda per lettura ottica (tipo modulo INPS)

Esempi di livelli
- Esempio di livello logico
  - Archivio anagrafico dell'ufficio del personale di un'azienda
- Esempio di livello esterno
  - Organigramma di un'azienda partendo dall'archivio dell'ufficio del personale
- Esempio di livello concettuale
  - Modello delle entità e delle loro relazioni all'interno di un'azienda astratta

Indipendenza tra livelli
- Indipendenza fisica
- Indipendenza logica

Modelli logici
- Gerarchico (da approfondire il concetto)
- Reticolare (da approfondire il concetto)
- Relazionale
- Ad Oggetti

Modello ad oggetti
- Oggetto
- Attributi
- Metodi
- Gerarchia ed ereditarietà

Modello relazionale

Storia

Proposto da E. F. Codd nel 1970 per favorire l'indipendenza dei dati
Disponibile in DBMS reali nel 1981 (non è stato facile implementativamente coniugare l'indipendenza con efficienza e affidabilità!)
Si basa sul concetto matematico di relazione (con una variante per renderle più "usabili")
Le relazioni hanno come naturale rappresentazione le tabelle

3.1 Relazioni e tabelle

Accezioni di Relazione

matematica: come nella teoria degli insiemi
informatica: relationship tradotto anche con associazione o correlazione
relazione secondo il modello relazionale dei dati

3.1.1 Relazione matematica

Esempio semplice di relazione a due domini

- **D1={a,b}**
- **D2={x,y,z}**
- **prodotto cartesiano D1 × D2**
- **relazione r ⊆ D1 × D2**

Definizione più generale di Relazione

Siano D1, …, Dn (n insiemi anche non distinti)
Definiamo prodotto cartesiano D1×…×Dn:
l'insieme di tutte le n-uple (d1, …, dn) tali che d1∈D1, …, dn ∈ Dn
Definiamo relazione matematica su D1, …, Dn:
un sottoinsieme di D1×…×Dn.
D1, …, Dn sono definiti come i domini della relazione

Una relazione matematica è un insieme di n-uple ordinate:
(d1, …, dn) tali che d1∈D1, …, dn ∈ Dn

Una relazione matematica è un insieme per il quale :

- non c'è ordinamento fra le n-uple
- le n-uple sono distinte
- ciascuna n-upla è ordinata: l' i-esimo valore proviene dall' i-esimo dominio
- la struttura è quindi detta posizionale

3.1.2 Relazione secondo il modello “relazionale”

Nel modello relazionale la struttura delle relazioni è non posizionale, ed ad ogni dominio della n-upla viene assegnato un nome che lo identifica in modo univoco all'interno della relazione.
Il dominio viene quindi chiamato “attributo”.

3.1.3 Relazioni – Relationsip

In informatica il termine Relationship indica una relazione tra due o più entità (da questo nasce poi un pericoloso rischio di confusione sul termine relazione in italiano)

La relationship è un concetto fondamentale in tutto ciò che è trattamento e rappresentazione della informazione in quanto la realtà è intrinsecamente composta da entità (oggetti) e dalle interazioni tra tali entità (come vedremo nel sistema di modellazione “entità-relazioni”)

3.1.4 Tabelle

Tabella è una struttura bidimensionale dove ogni colonna è caratterizzata da un nome univoco, ed ogni riga rappresenta una n-upla

Tutte le relazioni matematiche possono essere rappresentate come tabelle
Non tutte le tabelle sono rappresentazioni di relazioni matematiche

Una tabella rappresenta una relazione se
- le righe sono diverse fra loro
- le intestazioni delle colonne sono diverse tra loro
- i valori di ogni colonna sono fra loro omogenei

3.1.5 Rappresentazione delle relationship

Nel modello gerarchico ed in quello reticolare la relationship viene espressa tramite i puntatori che da un “record” permettono di referenziarne un altro (della stessa o di un'altra tabella)

Nel modello relazionale la relationship viene espressa tramite l'uguaglianza dei valori tra un sottoinsieme definito degli attributi di due “record” di due relazioni (siano esse anche eventualmente la stessa)

3.2 Analisi di dettaglio del modello Relazionale

Il modello relazionale è basato su una teoria matematica rigorosa, mentre i modelli precedenti erano il risultato dell'evoluzione implementativa

Concetti centrali del modello Relazionale
- Relazione
- Attributo
- Record o Tupla (Ennupla)

3.2.1 Definizioni matematiche

Schema di una relazione:

$R(A1,..., An)$

Schema di base di dati:

$R = \{R1(X1), ..., Rk(Xk)\}$

Base di dati : insieme di relazionii

Ennupla (in inglese T-upla) :

funzione che associa ad un insieme di attributi X una serie di valori appartenenti ai singoli domini dei vari attributi

oppure anche insieme di attributi valorizzati secondo i vincoli dei domini corrispondenti

valore di un attributo della ennupla:

il valore dell'attributo A per la ennupla t si indica con $\mathbf{t[A]}$

Relazione (o istanza di una relazione)

dato uno schema $R(X)$ si dice relazione un insieme r di ennuple appartenenti ad X

Base di Dati (o istanza di base di dati)

dato un insieme R di schemi Ri(Xi) indicabile con $R= \{R1(X1), ..., Rn(Xn)\}$ si definisce Base di Dati un insieme r composto da r_i relazioni su R_i, espresso matematicamente con $r = \{r1,..., rn\}$

3.2.2 Considerazioni sul modello

Vantaggi del modello relazionale:
- indipendenza dall'implementazione fisica
- vengono rappresentate solo le informazioni rilevanti dal punto di vista dell'applicazione
- i programmi e gli utenti ragionano sugli stessi dati
- i dati sono più facilmente portabili da un sistema ad un altro
- si supera la direzionalità dei puntatori dei modelli gerarchici e reticolari

Svantaggi:
- i puntatori sono più performani nella navigazione attraverso la base di dati

3.2.3 Attributi

Tipizzazione degli attributi

Esempi di tipi
- Numero
  - Intero
  - Decimal
  - Float
- Caratteri
  - Singolo carattere
  - Array di caratteri
  - Stringa di caratteri
- Tempo
  - Date
  - Time
  - Timestamp
- Altri tipi
  - Booleani
  - CLOB / BLOB
- Tipi composti ed “User Defined”

3.2.4 L'informazione incompleta – NULL

Cosa significa non avere il valore di un attributo
- non esiste quel valore
- il valore esiste ma non è noto
- il valore non è rappresentabile con quella struttura dati (pi-greco)
- etc..

Perché è necessario avere un valore NULL

Non conviene usare valori del dominio come "Jolly" non utilizzati perché:
- i valori Jolly potrebbero diventare non più non utilizzati
- è sempre necessario interpretare il valore secondo una codifica non esplicita

Non sempre ad un attributo A di una relazione R può essere concesso di assumere il valore NULL: è necessario definire dei "vincoli" sul modello

3.2.5 I Vincoli

Definizione di vincolo:
> restrizione imposta alle possibili ennuple di uno schema di basi di dati affinché tali tuple modellino in modo corretto una realtà specifica

Cioè i vincoli rappresentano i confini imposti dalla realtà alle infinite combinazioni dei dati insiemistici e matematici.

Oppure anche un vincolo è una proprietà che deve essere soddisfatta dalle istanze che rappresentano informazioni corrette per l'applicazione
E' sempre quindi relativo ad un'applicazione, non ad un dominio del possibile matematico.

I vincoli sono di diversi tipi :
- intrarelazionali (relativi ai valori di una singola ennupla o del singolo record)
  - di dominio (che riguardano i valori ammissibili per un certo attributo)
  - di ennupla (che legano tra loro i valo i dei vari domini di una ennupla)
- relazionali (che coinvolgono una relazione nella sua interezza)
- interrelazionali (cioè tra ennuple di diverse relazioni)

I vincoli intrarelazionali hanno la caratteristica di poter essere valutati considerando soltanto la singola ennupla, mentre quelli interrelazionali richiedono la valutazione di più relazioni contemporaneamente.

I vincoli relazionali hanno invece la caratteristica di riguardare una singola relazione, ma coinvolgono tutte le ennuple di tale relazione, possono essere visti come un caso degenere di interrelazionale.
Ma pur essendo un caso "degenere" hanno un'importanza particolare proprio nel modello relazionale per la definizione delle CHIAVI delle relazioni

3.2.6 Chiavi ed identificazione delle ennuple

Definizione intuitiva di chiave di una relazione:
> insieme di attributi che identifica in modo univoco le ennuple di una relazione

Definizione matematica di chiave:
- sia data una relazione r

- un insieme K di attributi è detto superchiave per r se r non contiene due ennuple distinte t1 e t2 per le quali t1[K] = t2[K]
- K è detto chiave per r se non esiste un sottoinsieme Ki di K che sia a sua volta superchiave di r (cioè K è una superchiave minimale di r)

Spesso si rischia analizzando solo i dati di esempio di una relazione di identificare come chiave un set di attributi che in realtà (cioè nel mondo reale che quello schema di relazioni vuole modellare) non ha tale caratteristica.

ATTENZIONE: nei DBMS relazionali non è necessario che per ogni tabella (relazione) sia definita una chiave, ma in tal caso non sarà sempre possibile operare sulla singola ennupla in quanto potrà capitare di avere in una tabella una serie di ennuple tra loro indistinguibili.
Questo non è di solito un bene, ma bisogna vedere cosa in realtà quello schema modella.

3.2.7 Esempi di rappresentazione di informazioni nel modello relazionale

Codifica delle informazioni (tabelle di trascodifica)

Strutture nidificate (o master-detail)

Strutture ricorsive (gerarchia, albero, grafo comunque connesso)

Algebra e calcolo relazionale

Introduzione ai linguaggi

Il DBMS è una scatola nera, ma dobbiamo interagire con esso.

Per interagire con il DBMS vengono definiti dei linguaggi per le Basi di Dati

Che tipo di operazioni possono essere fatte sul DBMS ?

- Interrogazione sui dati (Query)
- Manipolazione dei dati
- Definizione del modello dei dati

Funzioni dei linguaggi:

Interrogazione e manipolazione dei dati :

DML: data manipulation language

Definizione del modello dei dati :

DDL: data definition language

Tipologie dei linguaggi:

Dichiarativi - specificano le proprietà del risultato ("che cosa")
Procedurale - specificano le modalità di generazione del risultato ("come")

Esempi di linguaggi di interrogazione per DB:

Algebra relazionale – matematico di tipo procedurale basato sull'insiemistica
Calcolo relazionale – teorico matematico di tipo dichiarativo
SQL (Structured Query Language) – reale e solo parzialmente dichiarativo
QBE (Query by Example) – reale di tipo dichiarativo
Datalog – reale di programmazione logica basato su regole ed inferenza

L'Algebra relazionale

L'Algebra relazionale è una formulazione rigorosa e matematica che permette di operare sulle relazioni, dove le relazioni vengono viste come insiemi di n-uple manipolate attraverso operatori appunto insiemistici.

Si basa su:

- una serie di operatori insiemistici
- che operano sulle relazioni
- che possono essere tra loro combinati
- e che producono come risultato delle altre relazioni

Unione, intersezione, differenza

Sono le tre operazioni base dell'insiemistica ed hanno una corrispondenza diretta con gli operatori della logica booleana.

Sono operazioni binarie, cioè si applicano ad una coppia di relazioni (o insiemi)

Si possono applicare solo a relazioni che hanno lo stesso insieme di attributi o domini

Questa limitazione è di tipo matematico ma anche logico: non si sommano mele con pere

Il risultato di queste operazioni è sempre una relazione il cui schema è a sua volta uguale a quello delle due (o più) relazioni di partenza.

Unione : tutti gli elementi di un insieme o dell'altro – **OR - U**

Intersezione : tutti gli elementi di un insieme e dell'altro – **AND - ∩**

Differenza : tutti gli elementi di un insieme che non sono anche dell'altro - **NOT** - **-**

Queste operazioni possono essere tra loro liberamente combinate ed esistono diversi modi per ottenere lo stesso risultato, seguendo le varie proprietà associative, commutative, etc. ad esempio:

$$r \cap (s \cup t) = (r \cap s) \cup (r \cap t)$$

In particolare l'intersezione può essere definita a partire dalla differenza :

$$r \cap s = r - (r - s)$$

Quindi l'insieme minimo di operandi è dato da Unione e Differenza.

Selezione

La selezione è un operatore “monadico”, cioè che si applica ad una sola relazione

Produce una relazione il cui schema è identico a quello della relazione di partenza (detta “operando”)

La relazione ottenuta tramite una **selezione** contiene un sottoinsieme delle n-uple dell'**operando**, in particolare tutte e sole quelle che soddisfano una **condizione** definita dalla selezione stessa.

La selezione ha quindi tre elementi:

1. operando (relazione di partenza)
2. risultato (relazione risultante)
3. condizione (regola o funzione boolena che debbono soddisfare le n-uple dell'operando per appartenere alla relazione risultante

Sintassi della selezione: $\mathrm{SEL}_{Condizione}(Operando)$

Una selezione conterrà al massimo lo stesso numero di n-uple dell'operando, ma tendenzialmente meno in funzione del livello di “ristrettezza” delle condizioni applicate.

Proiezione

La selezione è anche lei un operatore "monadico".

Produce una relazione il cui schema è un sottoinsieme di a quello ella relazione di partenza (detta "operando")

La relazione ottenuta tramite una **proiezione** contiene al massimo un numero di n-uple pari a quello delle n-uple dell'operando.

La proiezione ha tre elementi:

4. operando (relazione di partenza)
5. risultato (relazione risultante)
6. lista degli attributi dell'operando che devono appartenere alla relazione risultante

Sintassi della proiezione: PROJ *ListaAttributi* (*Operando*)

E' importante fare delle considerazioni sulla cardinalità del risultato di una proiezione.

ATTENZIONE:

parliamo di relazioni MATEMATICHE, quindi che NON contengono n-uple uguali.

La relazione ottenuta tramite una **proiezione** contiene al massimo un numero di n-uple pari a quello delle n-uple dell'operando, ma può contenerne meno se:

la lista degli attributi della proiezione non è una superchiave dell'Operando

Se X è superchiave di R, allora numerosità (PROJx(R)) = numerosità (R)

Cioè

| PROJ x (R) | = | R | se x superchiave di R

Considerazioni su Selezione e Proiezione

La selezione e la proiezione possono essere viste come due operazioni tra loro "ortogonali".

Infatti :o

- la selezione "taglia" delle fette orizzontali di una tabella (appunto seleziona delle n-uple tra tutte quelle della relazione di partenza)
- la proiezione "taglia" delle fette verticali della stessa tabella (selezionando solo alcune delle colonne

La selezione e la proiezione possono essere combinate per ottenere delle porzioni particolari di una tabella (o relazione)

Ridenominazione

La ridenominazione è anche un operatore "monadico", che lascia inalterato il contenuto di una

relazione ma ne modifica lo schema cambiando il nome di alcuni attribuiti.

La funzione della ridenominazione è quella di permettere di trasformare due relazioni che hanno informazioni tra loro coerenti ma schemi i cui nomi differiscono, in modo da poter poi applicare su tali relazioni le altre operazioni dell'algebra relazionale.

E' importante quando due o più relazioni esprimono particolari tipologie di un'unica entità che viene in esse dettagliata secondo maggiori dettagli trascurabili nell'interrogazione in atto.

La sintassi di una ridenominazione è: $\text{REN } y \leftarrow x\ (R)$

Join

Il join (combinazione, aggregazione) è uno degli operatori più interessanti dell'algebra relazionale poiché permette di correlare dati tra relazioni diverse.

R_1 JOIN R_2 è una relazione su X_1X_2

$$\{ t \text{ su } X_1X_2 \mid \text{esistono } t_1 \in R_1 \text{ e } t_2 \in R_2$$
$$\text{con } t[X_1] = t_1 \text{ e } t[X_2] = t_2 \}$$

Cardinalità del JOIN

Il join di R1 e R2 contiene un numero di n-uple compreso fra zero e il prodotto di |R1| e |R2|

NATURAL JOIN

Il NATURAL JOIN è un tipo di operazione che ci permette di correlare due o più tabelle sulla base di valori uguali in attributi contenenti lo stesso tipo di dati.

INNER JOIN - Interno

È un tipo di join in cui le righe delle tabelle vengono combinate solo se i campi collegati con join soddisfano una determinata condizione.

OUTER JOIN - Esterno

Con l'OUTER JOIN è possibile estrapolare anche quei dati, *appartenenti ad una delle tabelle*, che non verrebbero estrapolati nei tipi di join visti fino a questo momento. Infatti OUTER significa esterno; dati esterni al normale tipo di join.

Il join esterno completa, con valori nulli, le n-uple che verrebbero tagliate fuori da un join (interno).

Può essere di tre tipi:

- sinistro
- destro
- completo

Join e proiezioni

Prodotto cartesiano

theta-join

$$R_1 \text{ JOIN}_{\text{Condizione}} R_2$$

Equivalenza di espressioni

Push selections (se A è attributo di R_2)

SEL A=10 (R1 JOIN R2) = R1 JOIN SEL A=10 (R2)

Selezione con valori nulli

Viste (relazioni derivate)

Viste virtuali e materializzate

Viste e aggiornamenti

Conclusione Algebra relazionale

Operatori dell'algebra relazionale

- unione, intersezione, differenza
- selezione
- proiezione
- ridenominazione
- join (join naturale, prodotto cartesiano, theta-join)

Calcolo relazionale

Calcolo e algebra

Il Calcolo relazionale fa parte di una famiglia di linguaggi dichiarativi, basati sul calcolo dei predicati del primo ordine.

Calcolo su Domini

Questo tipo di calcolo è composto da espressioni applicate a domini.

Le espressioni hanno la forma:

$$\{ A_1: x_1, \ldots, A_k: x_k \mid f \}$$

dove:

f e' una formula (che può utilizzare connettivi booleani e quantificatori)

$A_1: x_1, \ldots, A_k: x_k$ è una lista di coppie attributo – variabile, detta "target list", con

$A_1, \ldots, A_k$ attributi distinti (che possono anche non far parte della base di dati)

$x_1, \ldots, x_k$ variabili tra loro distinte

Il significato o risultato di un'espressione e' una relazione sui domini $A_1, \ldots, A_k$ che contiene le n-uple di valori per $x_1, \ldots, x_k$ tali per cui la formula f risulta verificata

Nelle formule possono essere utilizzati i qualificatori universali ed esistenziali:

- per ogni $\forall$
- esiste $\exists$

Il calcolo sui domini ha il difetto di essere molto lungo (per scrivere una semplice interrogazione ci vogliono predicati molto lunghi, e quindi è di difficile lettura) e soprattutto permette di scrivere espressioni senza senso.

Calcolo su ennuple con dichiarazioni di range

Per superare i limiti del calcolo sui domini viene definito il "calcolo su ennuple con dichiarazioni di range", dove per semplificare e snellire la scrittura ci si può riferire ad un'intera n-upla utilizzando una singola variabile (invece della lista delle coppie attributo-variabile), ed i valori ed i domini vengono ristretti ai soli provenienti da una definita base di dati.

Le espressioni del "calcolo su ennuple con dichiarazioni di range" hanno la seguente forma:

$$\{ \mathit{TargetList} \mid \mathit{RangeList} \mid \mathit{Formula} \}$$

dove

- *RangeList* è la lista delle relazioni associate alle relative variabili d'uso
- *TargetList* è la lista degli attributi che verranno estratti dalla *RangeList*
- *Formula* è appunto una formula che deve essere soddisfatta dalle n-uple delle relazioni della RangeList affinché i loro valori entrino a far parte del risultato nella TargetList

Calcolo e algebra

Calcolo e algebra sono "equivalenti"

per ogni espressione del calcolo relazionale che sia indipendente dal dominio esiste un'espressione dell'algebra relazionale equivalente a essa

per ogni espressione dell'algebra relazionale esiste un'espressione del calcolo relazionale equivalente a essa (e di conseguenza indipendente dal dominio)

Limiti dell'algebra e del calcolo relazionale

Chiusura transitiva

Supervisione(Impiegato, Capo)

Per ogni impiegato, trovare tutti i superiori (cioè il capo, il capo del capo, e cosi' via)

Nell'esempio, basterebbe il join della relazione con se stessa, facendo però ad ogni passo un'opportuna ridenominazione, ma servirebbero potenzialmente infinite join e ridenominazioni

Chiusura transitiva, impossibile!

Non esiste in algebra e calcolo relazionale la possibilità di esprimere l'interrogazione che, per ogni relazione binaria, ne calcoli la chiusura transitiva

SQL – Concetti di base

Linguaggio e Standard

Structured Query Language

SQL copre sia DDL che DML

Nascita 1970 - Laboratorio IBM – S.Josè – California per DBMS System-R

Prima Standardizzazione 1980 – ANSI – ISO

scarso supporto a definizione schemi ed istanze

successiva introduzione della definizione dei vincoli d'integrità referenziale

1992 – SQL-92 riferito anche come SQL-2

Adozione completa del modello relazionale

definizione di 3 livelli d'implementazione

- entry
- intermediate
- full

1999 – SQL-1999

Modelli ad oggetti

Trigger ed external prcedure

2003 – SQL-2003

Eliminazione costrutti obsoleti

Nuove estensioni tra loro indipendenti JRT, XML, etc

Definiti come SQL-3

Nessun sistema commerciale implementa SQL-3, molti arrivano a SQL-2 intermediate

Definizione dei Dati

Domini Elementari

Caratteri

- character / char
- varchar (character varying)
  - character set

Tipi numerici esatti

- numeric e decimal
  - precision + scale
- integer e smallint

Tipi numerici approssimati

- float (precisione)
- real
- double precision

Date e tempo

- date
- time e timestamp (precisione) (with time zone)

Intervalli temporali

- interval (unita' di tempo di massima) to (Unita' di tempo di dettaglio)

SQL-3

- boolean
- bigint
- BLOB e CLOB

Domini definiti dall'utente

Concetto di dominio definito dall'utente

Vantaggio per la consistenza e la manutenzione dei domini utente

Definizione

- CREATE DOMAIN *nomeDomain* as *TipoDiDato* [valoreDiDefault] [vincolo]

Schema - Database

Concetto di schema e di database

Definizione

- CREATE SCHEMA [nomeSchema]

```
CREATE DATABASE [nomeDatabase]
```

Tabelle

Creazione di una tabella

```
CREATE TABLE STUDENTI (
        ID int,
        NOME varchar(40),
        MATRICOLA char(10),
        DATA_NASCITA date,
        SESSO char(1) NULL,
        IN_CORSO char(1) DEFAULT 'S',
        PRIMARY KEY (ID) )
```

Vincoli

Vincoli Intrarelazionali

- null e not null
- unique
- primary key

Vincoli Interrelazionali

- Foreign key
  - on update
  - on delete
    - cascade
    - set null
    - set default
    - no action

Creazione di una vincoli interrelazionali

```
CREATE TABLE CORSI (
        ID_CORSO int,
        CODICE char(5),
        TITOLO varchar(40),
        DESCRIZIONE text,
        DATA_INIZIO date,
```

```
    CREDITI int DEFAULT 2,
    ANNO int ,
    REF_PROF int,
    PRIMARY KEY (ID_CORSO),
    UNIQUE INDEX IDX_CORSI_CODICE (CODICE)
    );

CREATE TABLE ESAMI_STUDENTE (
    ID_ESAME_STUDENTE SERIAL,
    RIF_STUDENTE int
        REFERENCES STUDENTI(ID),
    ID_CORSO int,
    DATA_ESAME date,
    VOTO int,
    PRIMARY KEY (ID_ESAME_STUDENTE),
    UNIQUE INDEX IDX_ESAMI_STUDENTE
        (RIF_STUDENTE, ID_CORSO, DATA_ESAME),
    CONSTRAINT FOREIGN KEY FK_CORSI (ID_CORSO)
        REFERENCES CORSI(ID_CORSO)
    ) ;
```

Definizione alternativa del vincolo tra ESAMI_STUDENTE e STUDENTI

```
ALTER TABLE ESAMI_STUDENTE ADD CONSTRAINT FOREIGN KEY FK_STUDENTI
(RIF_STUDENTE) REFERENCES STUDENTI(ID) ;
```

Modifica degli schemi

Alter di una tabella o di un dominio

ALTER TABLE nomeTabella

ALTER COLUMN nomeAttributo

ADD COLUMN nomeAttributo

DROP COLUMN nomeAttributo

ADD CONSTRAINT nomeConstraint

DROP CONSTRAINT nomeConstraint

Estensioni in SQL-3

Drop di una tabella

DROP TABLE nomeTabella (nomeDomain / nomeSchema / .....)

RESTRICT

CASCADE

Cataloghi relazionali

Concetto di metadati

Catalogo interno DEFINITION SCHEMA

Catalogo esterno INFORMATION SCHEMA

Manipolazione dei Dati

CRUD : Create, Read, Update, Delete

Create - Insert

Forma base :

INSERT INTO *Tabella* [(*Attributi*)] VALUES (*Valori*) [, (*Valori*)]

Opera su una sola tabella.
La lista dei valori è legata attraverso una relazione posizionale con la lista degli attributi.
Se la lista degli attributi non è fornita, si assume l'intera totalità degli attributi della tabella secondo l'ordine che hanno nella tabella stessa (ordine con cui sono stati creati) .

Rispetto dei vincoli di tabella.

Se un attributo non viene valorizzato (non è citato nella lista degli attributi e non compare quindi neppure in quella dei valori):
a – valore nullo
b – valore di default

Il comportamento dipende dal tipo di RDBMS utilizzato.
MySQL in caso di valore non indicato per un campo not – nullable inserisce un suo valore di default che non è indicato nella CREATE della tabella.

Read - Select

Forma base :

SELECT *ListaAttributi* FROM *ListaTabelle* [WHERE *Condizione*]

Tre elementi:

1. Cosa - SELECT (lista degli attributi da estrarre)
2. Da dove – FROM (lista delle tabelle sorgenti dei dati)
3. In che caso – WHERE (condizione di selezione) - Opzionale

Corrispondenza con Algebra Relazionale in caso di tabella singola

PROJ *ListaAttributi* (SEL *Condizione* (*Tabella*))

Primi casi particolari:

SELECT su unica tabella

SELECT * .....

Select senza proiezione

SELECT ... FROM .... senza WHERE – Prodotto cartesiano

Ordinamento

SELECT ..... ORDER BY ....

Parole chiave ASC (default) e DESC

Ordinamento su più campi

Gestione dei valori Nulli

- nella condizione
- nell'ordinamento
- nell'inserimento

Modificatori della molteplicità dei risultati:

DISTINCT (o DISTINCTROW)

ALL (default)

Funzione LIKE

Caratteri speciali per la ricerca:

_ : singolo carattere

%: quantità di caratteri a piacere

Espressioni nella lista degli attributi (proiezione)

Abbreviazioni (rinominare) : "AS"

Utilizzo di nomi (brevi) per indicare elementi della SELECT, siano campi o intere tabelle

FUNZIONI di aggregazione

Contatore dei record

Select count(*) from .....

COUNT (DISTINCT ....)

Count e valori nulli

Funzioni matematiche:

SUM, MIN, MAX, AVG, STD

Raggruppamento:

GROUP BY

Select su più tabelle

Determina il prodotto cartesiano delle varie tabelle

Necessario ridurre il prodotto cartesiano definendo delle condizioni

La condizione fondamentale è quella di legame tra le varie tabelle

Con la condizione di legame ottengo una JOIN

Identificazione dei campi tramite "dot notation" per disambiguare la query

SELECT … FROM *TabellaA* , *TabellaB* [WHERE *CondJoin* [AND *AltreCond*]] ....

JOIN Esplicito

SELECT … FROM *TabellaA* { … JOIN *TabellaB* ON *CondJoin* } … [WHERE *AltreCond*]

Le join possono essere concatenate, sia quelle esplicite sia quelle definite come prodotto cartesiano seguito da condizioni di Join.

Abbiamo diverse tipologie di JOIN:

NATURAL

THERA (definito senza parole chiave)

LEFT

RIGHT

FULL (non supportato da mySQL)

SELECT nidificati

Nella parte delle condizioni di una SELECT si possono utilizzare altre select per identificare particolari insiemi di valori accettabili.

Operatori Esistenziali

SELECT … FROM *TabellaA* WHERE *CampoA* in (SELECT ....)

SELECT … FROM *TabellaA* WHERE *CampoA* = ANY (SELECT ....)

SELECT … FROM *TabellaA* WHERE EXISTS (SELECT .... *CondizDiUnione*)

Visibilità delle variabili nelle query nidificate.

UNION

Unione tra i risultati di due o più select.

Eliminazione dei duplicati (a meno di modificatore **ALL**)

Campi possono non essere omogenei, creando strani effetti

EXCEPT

Differenza tra due select omogenee sulla stessa struttura

Esprimibile attraverso sotto-query o select nidificate. (non supportato da mySQL)

INTERSECT

Intersezione tra i resultset di due select, esprimibile attraverso opportune join e clausole condizionali.

(non supportato da mySQL)

DUAL

Tabella “virtuale” per poter fare interrogazioni su info non legate ad una tabella reale

HAVING

Possibilità di porre condizioni su valori risultanti della query (e non sui valori di partenza)

Utile in caso di raggruppamenti

Create – Insert - 2

Forma con Query :

INSERT INTO *Tabella* [(*Attributi*)] SELECT .....

Vengono inseriti in tabella i valori ottenuti dalla SELECT
La Select deve produrre un set di attributi coerente con la lista di attributi definiti nella INSERT o con la struttura della Tabella.

Update

UPDATE *NomeTabella*
SET *Attributo* =
< *Espressione* |
SELECT … |
NULL |
DEFAULT >
[, *Attributo* = ...]
[WHERE *Condizione*]

Se la Condizione viene omessa allora verranno aggiornate tutte le tuple della relazione.
Se esistono vincoli d'integrità referenziale l'aggiornamento potrebbe non andare a buon fine oppure scatenarsi l'aggiornamento dei valori anche di altre tabelle.

Delete

DELETE FROM *Tabella* [WHERE *Condizione*]

Elimina le tuple della tabella che soddisfano la condizione indicata
Se la condizione non è definita allora vengono eliminate tutte le tuple della relazione
Se esistono vincoli di integrità referenziale:

- potrebbe non essere possibile cancellare le tuple
- potrebbe scatenarsi una cancellazione a cascata anche su altre tabelle

Caratteristiche Avanzate

Vincoli d'integrità

FOREIGN KEY

Definito con la creazione della tabella:

CREATE TABLE tbl_name [(create_definition,...)]
Dove

create_definition:

col_name type [NOT NULL | NULL] [DEFAULT default_value]
[[PRIMARY] KEY] [reference_definition]
| PRIMARY KEY (index_col_name,...)
| KEY [index_name] (index_col_name,...)
| INDEX [index_name] (index_col_name,...)
| UNIQUE [INDEX] [index_name] (index_col_name,...)
| [CONSTRAINT symbol] FOREIGN KEY [index_name] (index_col_name,...)
[reference_definition]
| CHECK (expr)

Aggiunto successivamente ad una tabella:

ALTER [IGNORE] TABLE tbl_name alter_specification
Dove
alter_specification:

ADD [CONSTRAINT symbol] FOREIGN KEY [index_name] (index_col_name,...)
[reference_definition]

Altri vincoli

CHECK - relativo ad un campo od ad una intera tabella
(NON SUPPORTATO DA MYSQL 4)
PRIMARY KEY
UNIQUE INDEX
NOT NULL
ASSERT - relativo all'intero schema
(NON SUPPORTATO DA MYSQL 4)

I vincoli possono essere momentaneamente rilasciati durante una transazione, in modo che situazioni transitoriamente inconsistenti possono verificarsi, ma a fine transazione tutto deve essere coerente altrimenti la transazione viene annullata (NON SUPPORTATO DA MYSQL 4).

Viste

Significato ed uso di una vista:

create view *NomeVista* [(*ListaAttributi*)] as <*SelectSQL* >
[with [local | cascaded] check option]

Operazioni effettuabili su una vista:

- select
- update (con forti limitazioni – non sempre implementato)

Importanza progettuale delle viste:

- modello dei dati esterni
- indipendenza (parziale) dalle modifiche
- semplificazione concettuale delle interrogazioni

Definite anche le viste ricorsive in SQL 1999 ma non sono state implementate quasi da nessun DBMS

Funzioni Scalari

Servono alla manipolazione dei dati sia in selezione che in aggiornamento.

Operano sull'ennupla e restituiscono valori singoli

Tipologie:

- operatori di confronto (< , > , IS NULL, BETWEEN , IN , ...)
- operatori logici (NOT , AND , OR)
- controllo di flusso (IF , IFNULL , CASE)
- manipolazione stringhe (ASCII , BIN , CONCAT, LENGTH, LPAD, SUBSTRING,...)
- manipolazione date e tempi (DATE, TIME, DAYOFWEEK, MONTH, ADDDATE,...)
- conversioni (CAST,...)

Controllo dell'accesso

CHI – COME – COSA

Il sistema del controllo si basa su permessi o “privilegi”.

grant < *Privileges* | all privileges > on *Resource* to *Users* [with grant option]

dove i *Privileges* possono essere:

- insert: inserire nuovi record
- update: modificare il contenuto
- delete: eliminare record
- select: leggere i dati della risorsa
- references: definizione di vincoli di integrità referenziale verso questa risorsa
- usage: utilizzo in una definizione

Utilizzo di viste e permessi sulla stessa per filtrare i dati di una relazione e renderli solo parzialmente visibili ad un utente.

Nelle implementazioni più avanzate dell'SQL (non mySQL) esiste anche il concetto di ROLE (ruolo), a cui associare dei permessi.

Gli utenti vengono poi associati ai ruoli in modo da mantenere una coerenza di comportamento su un insieme di utenti.

Transazioni

Una transazione è un insieme di operazioni che devono essere eseguite in modo indivisibile.

Le proprietà dell'INDIVISIBILITA' sono (A.C.I.D.):

- Atomicità
- Consistenza
- Isolamento
- Persistenza (Durabilità)

Costrutti transazionali:

- start transaction (implicita)
- commit
- rollback
- autocommit

SQL nei linguaggi di programmazione

Sistemi informativi

I sistemi possono essere di almeno 3 tipi:

- Batch
- OnLine
- Service

Vi è la necessità far coesistere i programmi con i DB.

Due strade:

- espandere le potenzialità dei DBMS
- integrare nel linguaggi di programmazione l'SQL (in senso lato)

Stored procedure

Estende l'SQL con un linguaggio computazionalmente completo.

Le procedure vengono memorizzate all'interno del DBMS.

Resta il problema dell'interazione con l'utente.

Triggers

Realizzano il paradigma (o Pattern) Evento-Condizione-Azione (ECA).

Trigger quindi come azione da svolgere a fronte di un evento, se è valida una condizione.

Gli eventi possono essere:

- insert
- update (record o campo)
- delete

1 Trigger è associato (sensibile) ad 1 solo Evento.

L'azione può essere l'invocazione di una Stored Procedure !

Al trigger sono associati due elementi speciali: OLD e NEW, i record prima e dopo la modifica richiesta (nessun OLD per l'Insert, nessun NEW per il Delete).

I trigger possono permettere l'implementazione di vincoli d'integrità più sofisticati di quelli visti in precedenza.

Si possono realizzare catene di trigger, dove l'azione di uno causa lo scatenarsi di un altro trigger, e cosi' via, fino a generare situazioni estremamente complesse e col rischio di cicli infiniti (e conseguente blocco del sistema).

Linguaggi di 4 generazione

Embedded SQL

Meccanismo

Inserire del codice SQL all'interno di un programma scritto in un linguaggio normale.

Il programma verrà PRECOMPILATO da un apposito sistema che traduce le istruzioni SQL in chiamate a librerie particolari che permettono l'interazione tra il programma stesso e il DBMS specifico.

Vi è quindi la necessità di definire delle sezioni e delle parole chiave che permettono al precompilatore di capire cosa deve fare.

Problemi:

- specificità rispetto al linguaggio, al DBMS, all'ambiente (S.O., networking, etc.)
- "conflitto d'impedenza": programmi ragionano per record, SQL per insiemi

Statico

Le istruzioni SQL sono definite una volta per tutte, con al limite dei valori passati come parametri dal programma alle istruzioni SQL.

E' una soluzione che può avere ovviamente dei problemi di flessibilità, ma è molto performante perché il precompilatore può effettuare molte ottimizzazioni, ed inoltre molti errori possono essere scoperti in fase appunto di precompilazione.

Dinamico

Le istruzioni SQL sono costruite a run-time, ad esempio creando la stringa con pezzi diversi a seconda delle esigenze del momento.

Il precompilatore non può effettuare tutti i controlli e le ottimizzazioni del caso statico.

Se il programma non lavora bene possono verificarsi improvvisi errori dovuti a particolari situazioni di esecuzione.

Cursori

I cursori sono una soluzione per risolvere il problema del conflitto d'impedenza: permettono di scorrere (appunto cursore) lungo il set di dati ottenuti da una query.

Un cursore è associato quindi ad una select, e può essere impostato per essere mono o bidirezionale, per permettere o no aggiornamenti dei dati letti.

Fasi di vita di un cursore:

- DECLARE
- OPEN
- FETCH
- CLOSE

Se il cursore è mono-direzionale permetterà di leggere un record alla volta dal primo all'ultimo, attraverso il comando NEXT.

Invece se il cursore è bidirezionale permette di tornare indietro ma anche di saltare tra i vari record:

NEXT, PRIOR, FIRST, LAST, ABSOLUTE, RELATIVA

Call Level Interface (CLI)

Meccanismo

Definizione di una libreria (INTERFACE) che permette di effettuare delle chiamate (CAL) al DBMS tramite comandi SQL.

Questa libreria può essere più o meno generica, astratta rispetto a:

DBMS

Sistema Operativo

Networking

Se viene definito uno Standard, allora più sviluppatori, più società possono fare dei prodotti che sono tra loro INTERCAMBIABILI, cioè hanno la stessa INTERFACCIA.

Se il linguaggio ha poi un meccanismo per collegare dinamicamente diverse librerie che presentano la stessa interfaccia a seconda di opportuni parametri di configurazione allora i programmi avranno una portabilità elevatissima.

Esempio: JDBC

Elementi di un sistema che utilizza JDBC:

- applicazione Java
- JDBC Driver Manager
- Driver / Bridge / Middleware
- eventualmente Native Driver oppure ODBC
- DBMS

Struttura tipica di un programma che utilizza JDBC

- caricamento del driver (eventualmente sulla base di opportuni parametri di configurazione)
- apertura della connessione (con eventuali userId e password)
- creazione del contesto esecutivo (statement)
- esecuzione tramite lo statement delle istruzioni SQL (statiche o dinamiche)
- eventuale lettura dei dati attraverso oggetti di tipo ResultSet
- chiusura dello statement e della connessione

Progettazione di Basi di Dati

Ciclo di vita di un sistema informativo

•**Idea**: identificazione di una necessità da soddisfare e del come soddisfarla

•**Studio di fattibilità**: verifica dell'effettiva realizzabilità dell'idea, valutazione costi e priorità
•**Raccolta e analisi dei requisiti**: raccolta dei desideri e delle necessità a cui va incontro il sistema e studio delle sue proprietà
•**Progettazione**: di dati e funzioni
•**Realizzazione**: codifica delle funzioni progettate e creazione dell'infrastruttura applicativa
•**Validazione e collaudo**: verifica sperimentale del funzionamento del sistema
•**Funzionamento**: vita operativa del sistema
•**Evoluzione**: trasformazione del sistema sulla base delle nuove esigenze (interne od esterne)

Centralità dei dati e necessità della loro modellazione in una fase precoce.

Il dato è più stabile delle funzioni (ma non immutabile)

Ciclo di analisi e progettazione dei dati

- **ANALISI** (CHE COSA)
  - raccolta dei requisiti
  - progettazione concettuale
    - schema concettuale
- **PROGETTAZIONE** (COME)
  - progettazione logica
    - schema logico
  - progettazione fisica
    - schema fisico

Schemi e Istanze

schema: sostanzialmente invariante nel tempo, descrive la struttura (aspetto intensionale)
•nel modello relazionale, le intestazioni delle tabelle
istanza: i valori attuali, che cambiano anche molto rapidamente (aspetto estensionale)
•nel modello relazionale, il “corpo” di ciascuna tabella

Modelli

modelli concettuali: permettono di rappresentare i dati in modo indipendente dal sistema
•cercano di descrivere i concetti del mondo reale
•sono utilizzati nelle fasi preliminari di progettazione

modelli logici: utilizzati per l'organizzazione dei dati da parte dei DBMS
- •utilizzati dalle applicazioni
- •indipendenti dalle strutture fisiche

Motivi per l'uso dei modelli concettuali

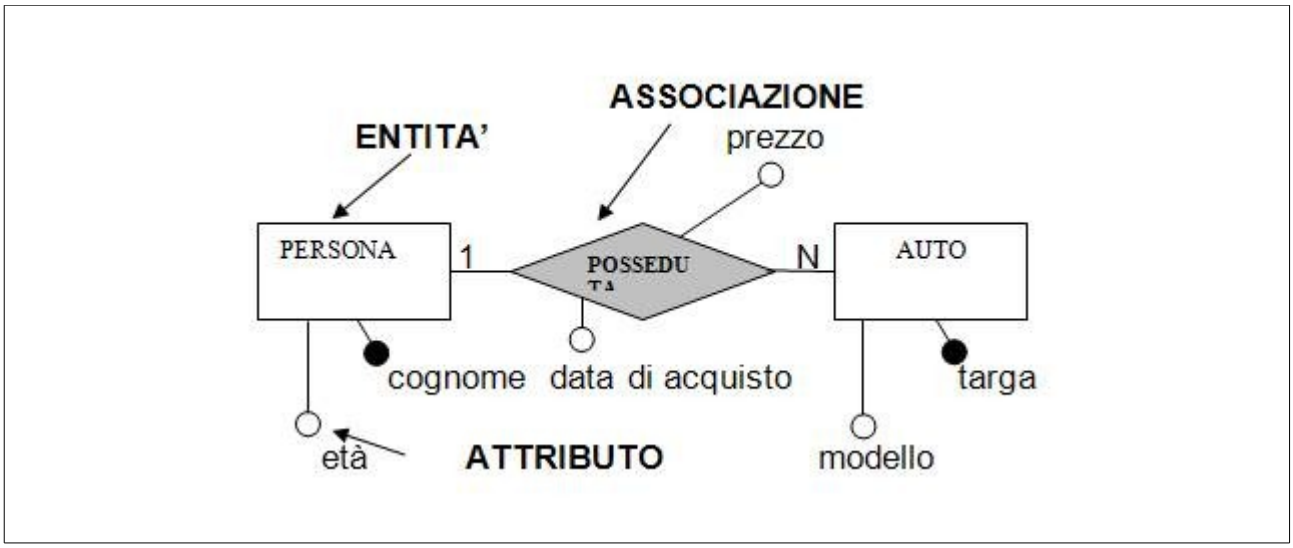
Se partiamo a progettare direttamente dallo schema logico della base di dati di un'applicazione:
•da dove cominciamo?
•rischiamo di perderci subito nei dettagli
•dobbiamo pensare subito a come correlare le varie tabelle (chiavi etc.)
•ci sono grosse rigidi tà

Il modello concettuale ha:
- una rappresentazione grafica (più comunicativa)
- maggiore astrazione rispetto ai dettagli implementativi
- quindi più semplice
- quindi più comprensibile per l'utente/committente del sistema

La tecnica di modellazione concettuale più diffusa è quella "**Entity-Relationship**" (E-R)

Modello Entity-Relationship

Elementi e costrutti del modello E-R:
- **Entità** - rappresentazione di una categoria di oggetti del mondo reale che viene modellato
- **Attributo** – caratteristica dell'entità d'interesse nel modello (aggregabili in attributi composti)
- **Relazioni o associazioni** (Relationship) – collegamento tra diverse entità del modello
- **Cardinalità** – delle relazioni ma anche degli attributi
- **Identificatore** – insieme di attributi e/o relazioni che identificano un'entità
- **Generalizzazione** – classificazione gerarchica delle entità con ereditarietà delle caratteristiche (attributi, relazioni, etc..)



Entità

Di solito si tratta di classi di oggetti (non singoli oggetti ma insiemi di oggetti con caratteristiche comuni) che possiedono un significato ed un'esistenza autonoma
Tutto dipende dal contesto applicativo, quella che può essere un'entità per un progetto potrebbe non esserlo in un altro (persone come entità in un sistema anagrafico, come valore numerico in un sistema geografico).
Ogni singola entità è indicata con un nome, espressivo all'intero del contesto applicativo.
Si seguono di solito delle opportune convenzioni, come quella di utilizzare nomi al singolare.

Attributi

Sono in prima approssimazione i campi di una tabella.
Possono anche essere **COMPOSTI.**

Relationship

Legame logico fra due o più entità, **rilevante** nell'applicazione di interesse
Il legame padre-figlio è fondamentale in un sistema per l'anagrafe civile, ininfluente in un sistema di gestione clienti.
Anche le relazioni hanno un nome, che sia possibilmente significativo e possibilmente non un verbo ("succedere", "contiene", ...) ma un sostantivo ("successione","componente", ...).
E' possibile definire per le varie entità che sono collegate da una relazione dei ruoli, che appunto dettagliano qualora sia necessario che parte gioca la singola entità nella relazione.

Tra due entità possono esservi più relazioni, ognuna con un suo particolare significato (ad esempio tra le entità "persona" e "città" possono esserci le relazioni "Residenza", "Nascita", "Domicilio",..).

Le relazioni possono essere tra più di due entità, ed in alcuni casi possono essere "opzionali" tra due o più entità.

Le relazioni possono essere anche ricorsive sulla stessa entità.
In questo caso di solito è necessario definire i ruoli all'interno della relazione che altrimenti potrebbe essere interpretata nel verso sbagliato (ad esempio per la relazione "genitore" sull'entità "persona" dev'essere indicato quale è la "madre" e quale il "figlio").

Promozione di una relazione
Un attento esame delle relazioni a volte porta a "promuovere" una relazione e trasformarla in un'entità.
Ad esempio un esame visto in un primo momento come una relazione tra studente e corso può poi diventare per esigenze applicative un'entità autonoma con una relazione con studente ed una con corso.
Di solito questo avviene quando ad una relazione iniziano ad essere associati molti attributi.

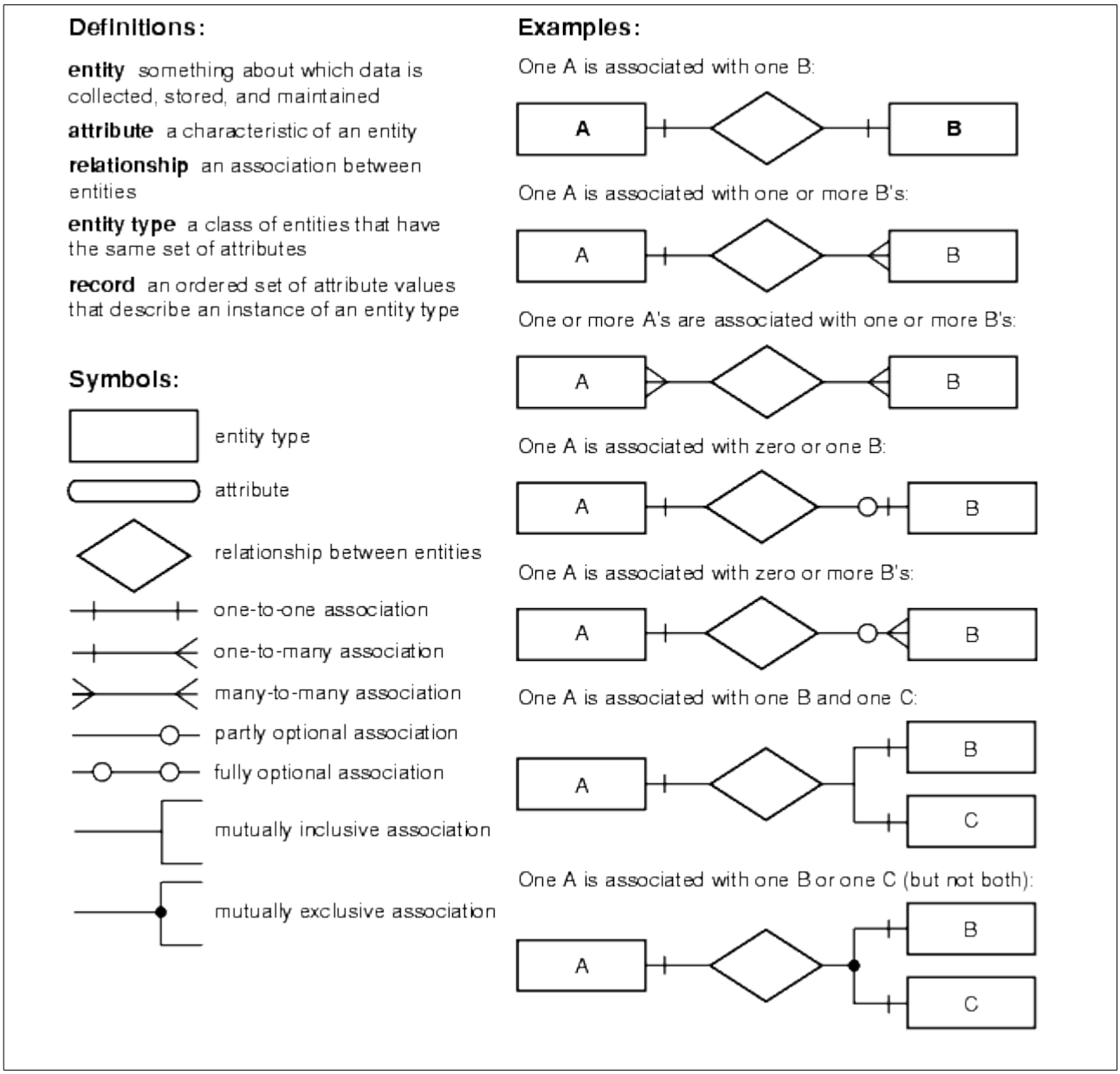
Cardinalità

Indica genericamente la numerosità di un ente.

Di due tipi:

- cardinalità di una relazione
- cardinalità di un attributo

La **cardinalità di una relazione** è una coppia di numeri interi che indica, per ogni "ramo" di una relazione (cioè per ogni entità che appare nella relazione), il numero minimo e massimo di volte che una relazione può esistere per una particolare occorrenza (oggetto) di un'entità.
Ad esempio la relazione ricorsiva "genitore" avrà una cardinalità 0-n per il ramo "madre" (cioè una persona può essere "madre" da 0 a n volte), mentre avrà una cardinalità 1-1 per il ramo "figlio" (cioè una persona deve sempre essere "figli" una ed una sola volta).



Identificatore

Insieme di attributi e/o di relazioni che identifica un'entità.
Gli identificatori sono i candidati a diventare le chiavi.

Generalizzazione

Una serie di entità potrebbero avere delle caratteristiche in comune, come ad esempio un impiegato,

uno studente, un professore: hanno tutte un nome, un cognome, una data di nascita, etc.

In questo caso si può pensare ad un'entità ”Persona” che è la generalizzazione di tutte queste entità e definire che le varie impiegato, studente e professore derivano da questa entità generale, ereditando da essa tutte le caratteristiche e definendone eventualmente di nuove.

Progettazione Concettuale

Raccolta dei requisiti

Dove si raccolgono i requisiti:

- dagli utenti del sistema
- nelle documentazioni relative al sistema
- da realizzazioni preesistenti (nello stesso contesto o in altri contesti)

Caratteristiche di una buona raccolta dei requisiti

- corretto livello di astrazione
- semplicità
- completamente esplicito (nulla di lasciato implicito)
- standardizzazione (dei termini e delle strutture)
- glossario

E' sempre e costantemente necessario verificare con l'utente la corretta comprensione da parte di entrambi e la consistenza e coerenza di tutto quanto descritto.

I requisiti non possono essere solo sui DATI, ma anche sulle OPERAZIONI da effettuare su tali dati, avendo quindi un approccio OLISTICO all'analisi.

Principio di indeterminatezza di Heisemberg (reciproco)

L'osservatore influenza il sistema osservato (ma anche il sistema osservato influenza l'osservatore).

Problema intrinseco della raccolta ed analisi dei requisiti: raccogliendo le informazioni se ne viene influenzati e quindi non si è più oggettivi, e interagendo con gli utenti non si è più neutrali.

Il rischio è conservare senza motivo caratteristiche dei sistemi e dei metodi esistenti senza metterli in discussione (e quindi compromettendo il nuovo sistema con eredità del vecchio), o essere prevenuti nei confronti di parti o utenti del sistema sulla base di quanto dettoci in precedenza.

Documentazione di progetto

Il modello E-R rappresenta in modo formale i concetti ma non li descrive, e non è adatto per la rappresentazione di vincoli complessi.

E' necessario integrare il modello E-R con una descrizione (formale o no) delle rimanenti parti del sistema.

Business Rule

Le business rule relative all'analisi dei dati sono di tre tipi:

- descrizione di concetti – in forma testuale libera
- dichiarazione di vincoli d'integrità
  - asserzioni del tipo: <ente del modello> **deve / non deve** <proprietà o espressione>

- derivazione di una proprietà da altre esistenti

  – operazioni aritmetiche o logiche per dedurre proprietà o enti da altre informazioni presenti nel modello, del tipo: <risultato> **si ottiene** <espressione o formula>

Vi sono poi altre business rule che meglio si adattano alla modellazione dei processi aziendali, descrivendo appunto le regole aziendali che governano la trasformazione dei dati, ed hanno la forma:

se <condizione iniziale> **allora** <descrizione del risultato di un processo di trasformazione>

Scenari d'uso

Uno strumento molto importante per l'analisi dei sistemi è la raccolta e la descrizione di scenari d'uso, cioè di precise sequenze di azioni ed eventi (come la sceneggiatura di un film) relativi a processi reali o da realizzare.

Sono utili soprattutto per la descrizione della parte dinamica del sistema, ma poiché i dati dovranno essere utilizzati all'interno di questo sistema dinamico, per ben progettare i dati devono essere noti anche i processi.

Permettono di far vedere (come in un film) all'utente come sarà il sistema, ma anche di formalizzare la descrizione dell'esistente per una più chiara comprensione da parte di progettisti ed analisti.

Progettazione

Strategie di progetto in generale:

- Top-down
- Bottom-up
- Inside-out
- Prototipale (non su libro)
- Ciclica (non su libro)
- eXtreame Programming (XP - non su libro)

Per la progettazione dei dati non si possono adottare soluzioni prototipali, ne tanto meno XP.

Per i dati la strategia è tipicamente top-down ciclica, cioè fasi di progettazione top-down da un riesame della correttezza e coerenza delle definizioni top sulla base dei risultati down.

Top-down

Modello più anziano, detto anche “a cascata”.

Presuppone la possibilità di completare una fase senza più necessità di riesame sulla base delle attività successive, quindi:

- perfetta conoscenza del dominio
- stabilità dei requisiti
- nessun effetto retroattivo tra dettaglio e generale (il dettaglio non nasconde nulla che non sia già definito a livello generale)

Bottom-up

Dopo una fase di “break-down” in cui il sistema viene spezzettato in elementi sempre più piccoli non ulteriormente scomponibili, si analizzano gli atomi per poi procedere ad aggregazioni successive fino ad ottenere il modello completo del sistema.

Aggregando i sistemi parziali si possono individuare:

- T1 - oggetti che hanno tutte le caratteristiche in comune (unificazione di entità)
- T2 - oggetti di sotto-sistemi diversi che hanno un legame (relazione)
- T3 - oggetti di sotto-sistemi diversi che sono casi particolari di un concetto più generale (generalizzazione)
- T4 e T5 - serie di attributi ricorrenti nei vari sotto-sistemi che possono essere aggregati in un'unica entità o relazione

E' un approcio da chimica-molecolare, che presuppone:

- la possibilità d'individuare subito tutte le componenti del sistema
- che il totale non abbia proprietà che trascendono la somma delle parti (problema della vita: come la si spiega partendo dalla sola chimica molecolare ?)

E' una ottima strategia per affrontare sistemi di grossissime dimensioni in cui un singolo analista non può essere in grado di dominare ogni dettaglio, ma l'integrazione dei modelli è sempre molto complessa ed è soggetta alla capacità di mediazione e di contrattazione tra i vari analisit.

Inside-out

Si procede dall'analisi completa di una porzione del sistema complessivo, per poi aggiungere altre parti da analizzare integrandole con quanto già modellato.

Adatta a sistemi che hanno un “core” (nucleo) molto ben definito e coerente, intorno al quale ruotano una serie di altri elementi lascamente correlati al nucleo.

Ad esempio la progettazione di un intero sistema ferroviario si focalizza prima di tutto sul treno e le rotaie, poi sulle stazioni e le altre infrastrutture.

Completata l'analisi si procede poi all'implementazione del sistema completo.

Non facilita una visione globale ed è poco astratta.

Prototipale (non su libro)

Quando il dominio non è noto, un approccio possibile è quello prototipale, che prevede la creazione appunto di un prototipo sul quale verificare i concetti di base e discutere le caratteristiche del sistema con l'utente, e poi sulla base di quanto raccolto procedere con una strategia classica di implementazione.

Ciclica (non su libro)

Si procede come nell'Inside-out, ma spingendo oltre l'analisi fino all'implementazione completa della parte, per poi fare nuovi cicli di analisi e implementazione di parti sempre più ampie del sistema.

eXtreame Programming (XP - non su libro)

Considerazioni sui costi e sugli aspetti di progettazione, realizzazione e produzione per sistemi: manifatturieri, informatici, di definizione dei dati.

- *Prodotti manufatturieri: peso preponderante sulla fase di produzione di serie (anche se aumentano sempre più i costi di progettazione)*
- *Prodotti informatici: peso quasi paritetico tra progettazione e produzione del primo singolo manufatto (costo di produzione di serie asintoticamente nullo)*
- *Modello dati: peso quasi completamente sulla parte progettuale (il processo di produzione è quasi sempre automatizzabile con strumenti CASE e comunque il prodotto è il risultato della pregettazione)*

Schema concettuale

Qualità di uno schema concettuale

- correttezza
- completezza
- leggibilità
- minimalità

Strumenti di progettazione assistita (CASE)

Progettazione Logica

Fasi della progettazione logica:

- analisi delle prestazioni
- ristrutturazione degli schemi
- traduzione in modello relazionale

Non si tratta di un processo lineare in tre passi ma di una serie di cicli di raffinamento, soprattutto tra i primi due passi.

Infatti una particolare ristrutturazione può richiedere una nuova verifica delle prestazioni per controllare il raggiungimento e/o il mantenimento degli obiettivi di performance desiderati, ed un'analisi delle prestazioni insoddisfacenti può scatenare una ristrutturazione dello schema con lo scopo di migliorarle.

Analisi delle prestazioni

L'analisi delle prestazioni si basa sulla valutazione del costo di un'operazione in termini di accessi al DB e di memoria necessaria.

Alla base vi è la descrizione del modello dati e l'individuazione delle attività svolte dal sistema.

E' necessario quindi avere:

- volume dei dati del modello (numerosità e dimensione media)
- caratteristiche delle operazioni
  - tipologia (batch / onLine)
  - frequenza
  - struttura dell'operazione (le query coinvolte per poter capire quali sono i dati toccati)

Ristrutturazione degli schemi

Passi di ristrutturazione

- analisi delle ridondanze
- trasformazione delle generalizzazione
- eliminazione attributi multi-valore
- accorpamento entità (relazioni 1-1)
- partizionamento verticale di entità (parte dei dati di un'entità non vengono quasi mai usati)
- identificatori principali

La ristrutturazione deve essere guidata di due obiettivi:

1. pulizia del disegno
2. ottimizzazione delle prestazioni

Questi due obiettivi spesso vanno di pari passo, ma a volte divergono, ed è quindi necessario trovare il giusto equilibrio tra “perfezione formale” del disegno e “sporca ottimizzazione” delle operazioni.

Traduzione in modello relazionale

Passi di traduzione

- entità
- associazioni uno a molti
- associazioni molti a molti
- associazioni uno a uno
- associazioni multiple
- identificatori esterni di entità

Progettazione Logica

Traduzione dello schema concettuale in uno schema logico che rappresenti gli stessi concetti e gli stessi dati in maniera *corretta* ed *efficiente.*

Fasi della progettazione logica:

- analisi delle prestazioni
- ristrutturazione degli schemi E-R
- traduzione in modello relazionale

Non si tratta di un processo lineare in tre passi ma di una serie di cicli di raffinamento, soprattutto tra i primi due passi.

Infatti una particolare ristrutturazione può richiedere una nuova verifica delle prestazioni per controllare il raggiungimento e/o il mantenimento degli obiettivi di performance desiderati, ed un'analisi delle prestazioni insoddisfacenti può scatenare una ristrutturazione dello schema con lo scopo di migliorarle.

Input della progettazione logica:

- schema concettuale
- analisi del carico operativo
    - descrizione delle operazioni principali del sistema
    - analisi dei volumi dei dati e delle operazioni
- tipo di modello logico da realizzare

Analisi delle prestazioni

L'analisi delle prestazioni si basa sulla valutazione del costo di un'operazione in termini di accessi al DB e di memoria necessaria.

Alla base vi è la descrizione del modello dati e l'individuazione delle attività svolte dal sistema.

E' necessario quindi avere:

- volume dei dati del modello (numerosità e dimensione media)
- caratteristiche delle operazioni
    - tipologia (batch / onLine)
    - frequenza
    - struttura dell'operazione (le query coinvolte per poter capire quali sono i dati toccati)

Su uno schema concettuale le prestazioni possono solo essere stimate.

Strumenti per la valutazione delle prestazioni:

- schema di navigazione dell'entità per la singola operazione
- da cui tavola degli accessi – con numerosità e tipologia di accesso

Ristrutturazione degli schemi E-R

Serve per:

- facilitare la successiva traduzione dello schema E-R in uno schema relazionale.
- ottimizzazione delle prestazioni

Passi di ristrutturazione

- analisi delle ridondanze
- trasformazione delle generalizzazione
- eliminazione attributi multi-valore
- accorpamento entità (relazioni 1-1)
- partizionamento verticale di entità (parte dei dati di un'entità non vengono quasi mai usati)
- partizionamento orizzontale di relazioni multiple
- identificatori principali

La ristrutturazione deve essere guidata di due obiettivi:

1. pulizia del disegno
2. ottimizzazione delle prestazioni

Questi due obiettivi spesso vanno di pari passo, ma a volte divergono, ed è quindi necessario trovare il giusto equilibrio tra "perfezione formale" del disegno e "sporca ottimizzazione" delle operazioni.

Analisi delle ridondanze

Le ridondanze creano problemi si allineamento dei dati, ma in alcuni casi possono velocizzare le interrogazioni.

Vantaggi:

- interrogazioni più semplici

Svantaggi:

- complicazione ed appesantimento degli aggiornamenti e **rischio di inconsistenza**
- maggiore occupazione di spazio

Le ridondanze possono essere dovute:

- attributi ripetuti tra entità
- attributi i cui valori possono essere altresì derivati da altri attributi della stessa entità
- attributi i cui valori possono essere derivati da attributi di altre entità
- attributi i cui valori possono essere derivati da conteggi su entità
- associazioni derivabili da altre associazioni

Trasformazione delle generalizzazioni

Le generalizzazioni possono essere trasformate secondo tre linee principali, in base alle tipologie di partizionamento delle informazioni tra entità generale ed entità di dettaglio, sia dal punto di vista

della completezza che dell'uso che ne viene fatto.

Le tre linee guida sono:

1. un'unica tabella che accorpa entità generale e tutte le tipologie di dettaglio (minimo comune multiplo, 1 sola tabella) – necessario metodo per identificare il tipo
2. replica della struttura del generale nelle varie entità di dettaglio (n tabelle)
3. entità separate per parte generale e per i vari dettagli (1 + n tabelle)

Ogni linea presenta vantaggi e svantaggi che vanno pesate sulla base dell'uso che si fa dei dati.

1.conviene se gli accessi al generale ed al dettaglio sono contestuali

2.conviene se gli accessi ai dettagli sono distinti per tipologia

3.conviene se gli accessi ai dettagli sono separati dagli accessi alle informazioni generali

Eliminazione attributi multi-valore

Gli attributi multivalore possono essere mappati con una tabella di dettaglio con una relazione uno a molti con quella di origine.

Intrinsecamente i campi multivalore hanno dei limiti dal punto di vista della numerosità massima gestibile e dell'accesso al singolo valore, problemi che possono essere superati trasformando appunto in una tabella di dettaglio che mantiene una dipendenza con l'entità principale.

Accorpamento entità (relazioni 1-1) - Partizionamento verticale di entità

Ogni volta che si presenta una relazione 1-1 tra due entità si deve valutare se tale relazione non nasconda in realtà un'identità tra le due entità coinvolte, cioè che le due entità modellate rappresentino aspetti diversi della stessa entità reale.

Dev'essere quindi valutata attentamente l'opportunità di accorpare in un'unica entità gli attributi delle due entità di partenza, tenendo presente che a volte considerazioni di carattere prestazionale possono spingere a non effettuare tale operazione.

Nel senso opposto le varie entità devono essere analizzate in funzione delle interrogazioni svolte su di esse in modo da ottimizzarne l'accesso.

Infatti se una parte dei dati non viene quasi mai acceduta può essere “parcheggiata” in una entità associata che pur mantenendo le informazioni non appesantisce le interrogazioni “abitudinali”.

Oppure se alcune interrogazioni si focalizzano su una parte di dati ed altre su una parte diversa, queste parti possono essere focalizzate “specializzando” due sotto entità in funzione delle interrogazioni.

Partizionamento orizzontale di relazioni multiple

Una relazione 1-n o n-m può avere caratteristiche di accesso particolari, cioè il legame può essere prioritario su una parte di questa relazione, soprattutto quando tale relazione descrive la storia di un'entità o di una relazione che varia nel tempo. Ad esempio una relazione tra un'automobile e tutti i suoi proprietari avrà una focalizzazione sul proprietario attuale, essendo gli altri temporalmente superati.

In tal caso si può prendere in considerazione l'opportunità di dividere la relazione originaria in due relazioni che rappresentano una la caratteristica prioritaria (ad esempio la situazione corrente) e

l'altra tutte le caratteristiche secondarie (ad esempio lo storico di tutte le relazioni).

Identificatori principali

Un passo fondamentale è l'analisi delle entità per l'individuazione degli identificatori principali (o chiavi) delle entità stesse.

I criteri di base per l'individuazione degli attributi adatti sono:

- assenza di opzionalità (cioè not null)
- minimalità (dannoso aggiungere attributi superflui)
- valutazione dell'utilizzo relativamente alle varie operazioni

Se nessun insieme di attributi soddisfa queste caratteristiche si può ricorrere alla creazione di nuovi attributi ad hoc: i **codici identificativi**.

Tali codici assumono valori generati appositamente, di solito attraverso **SEQUENCE**

Traduzione in modello relazionale

Passi di traduzione

- entità
- associazioni molti a molti
- associazioni uno a molti
- associazioni uno a uno
- associazioni multiple
- identificatori esterni di entità

Entità

Le entità si mappano direttamente in relazioni che hanno come attributi gli stessi dell'entità originante.

Associazioni molti a molti

Si trasformano in una relazione che ha come attributi le chiavi delle due entità che sono collegate da tale relazione, oltre agli attributi della relazione stessa.

L'unica limitazione è nella cardinalità della relazione poiché i valori minimi e massimi possono essere imposti solo attraverso complicati costrutti.

Associazioni uno a molti

L'associazione viene espressa come attributi nella relazione che partecipa con cardinalità maggiore, che referenziano le chiavi dell'altra entità, più gli eventuali attributi dell'associazione stessa.

Infatti date le due entità coinvolte A e B con cardinalità 1 ed n, l'associazione esisterà al massimo per ogni occorrenza dell'entità B, cioè per ogni tupla della relazione che mappa B.

Se l'associazione ha cardinalità minima 0, cioè non è obbligatoria, gli attributi di chiave e dell'associazione saranno nullable.

Associazioni uno a uno

Tale tipo di associazione può essere:

- fusa in una delle due entità (relazioni) coinvolte
- portare a fondere completamente anche le due entità dell'associazione

La scelta dipende dalle caratteristiche applicative di accesso ai dati.

Associazioni multiple

Le associazioni multiple vengono tradotto in modo simile a quanto fatto per le associazioni più semplici, con l'accortezza di identificare quali sono le relazioni che meglio si adattano ad essere usate come punto di aggregazione per le chiavi delle altre relazioni.

Identificatori esterni di entità

L'identificatore della relazione verrà definito aggiungendo anche l'identificatore della relazione esterna

Normalizzazione

Anomalie

Ridondanza
→ anomalia di aggiornamento
→ anomalia di inserimento
→ anomalia di cancellazione

Dipendenza Funzionale

•sia data una relazione ***r*** su ***R(X)***
•si considerino due sottoinsiemi non vuoti ***Y*** e ***Z*** di ***X***
•si dice che esiste in ***r*** una ***dipendenza funzionale*** (FD) da ***Y*** a ***Z*** se, per ogni coppia di tuple ***t1*** e ***t2*** di ***r*** aventi ***Y*** identico (cioè con gli stessi valori su Y), risulta che ***t1*** e ***t2*** sono identiche anche su ***Z*** (cioè hanno gli stessi valori anche su Z)

In altri termini si può dire che Y implica Z :

Y→ Z

Si hanno delle dipendenze funzionali BANALI quando
•Y → A con A appartiene a Y
Per simmetria
•Y → Z è non banale se nessun attributo in Z appartiene a Y

Una dipendenze funzionali può causare anomalie se la componente Y "di partenza" non corrisponde ad una chiave della relazione in esame

le FD non si "ricavano" dall'analisi dei dati,
ma ragionando sugli attributi dello schema

Forme normali

Si definisce che una base di dati è in Forma Normale (o soddisfa una certa forma normale) se sono rispettate delle proprietà che garantiscono l'assenza di determinati difetti strutturali.

Queste proprietà sono dette "Forme Normali".

Le "Forme Normali" sono quindi proprietà che si riferiscono ad una o più relazioni (intese in senso "relazionale") o a strutture ad esse equivalenti, e che indicano se tale relazioni garantiscono un determinato livello di consistenza dei dati.

Vengono definite 5+1 forme normali, che sono via via più stringenti.

Prima Forma Normale (1NF)

Una relazione è in 1NF se:

- non presenta attributi non atomici (ad esempio multipli)
- ha definita una chiave primaria

Tabelle devono descrivere entità singole - Una colonna un valore

Seconda Forma Normale (2NF)

Una relazione è in 2NF se:

- è in 1NF
- tutti i campi non in chiave dipendono funzionalmente dall'intera chiave primaria (e non da una parte di essa)

La chiave, solo la chiave, nient'altro che la chiave
2NF viene ottenuta spezzando tabelle in parti normalizzate che descrivano una singola entità

Terza Forma Normale (3NF)

Una relazione r è in 3NF se:

- è in 2NF
- per ogni dipendenza funzionale non banale $X \rightarrow Y$ definita su r
  - X contiene una chiave K di r
  - ogni attributo in Y è contenuto in almeno una chiave di r

Rimuovere colonne calcolate e creare tabelle di lookup

Vi è un importante teorema :

Ogni relazione può essere trasformata in modo da soddisfare la 3NF

Forma normale di Boyce e Codd (BCNF)

Una relazione r è in forma normale di Boyce e Codd se e solo se:

- per ogni dipendenza funzionale (non banale) $X \rightarrow Y$ definita su di essa, X contiene una chiave K di r

Si può dire anche che X dev'essere una superchiave di r

Se una relazione ha una sola chiave, allora essa è in BCNF se e solo se è in 3NF .

La BCNF garantisce un ottimo livello di consistenza dei dati, ma:
non sempre si può trasformare uno schema in modo che soddisfi la BCNF.

Quarta Forma Normale (4NF)

Una **base di dati** è in 4NF se:

- per ogni relazione di dipendenza funzionale molti a molti $X \rightarrow Y$, X è una superchiave

Quinta Forma Normale (5NF)

Una **base di dati** è in 5NF se:

- è in 4NF
- ogni relazione di dipendenza funzionale è implicata dalle chiavi delle relative tabelle

Processo di normalizzazione

La normalizzazione di un modello è il processo che mira ad individuare se e dove intervenire sul modello stesso al fine di eliminare attraverso la trasformazioni in una delle varie forme normali le anomalie, e con esse i rischi d'inconsistenza e ridondanza.

La normalizzazione non è una metodologia di progettazione delle Basi di Dati ma uno strumento di verifica.
La verifica pre normalizzazione mira ad identificare le anomalie che sono presenti in un modello relazionale e quindi i punti dove intervenire.

Più alto è il livello di normalizzazione raggiunto da un modello e maggiore è l'affidabilità dei dati in esso contenuti, ma maggiore è anche lo sforzo per gestire tali dati.

Uno schema in 4 o 5 NF ha un altissimo livello di rigore formale e garantisce un alto grado di non ridondanza e non anomalie, ma a costo di un'elevato degrado delle prestazioni, quindi non si utilizzano quasi mai, o meglio non si spinge il processo di normalizzazione di un modello fino al 4 o 5 NF.

La terza forma normale è meno restrittiva della forma normale di Boyce e Codd (e ammette relazioni con alcune anomalie) , ma ha il vantaggio di essere sempre “raggiungibile” .
Normalmente quindi ci si “accontenta” di un modello in 3NF.

Il processo di normalizzazione mira ad eliminare da un modello relazionale le anomalie identificate da una precedente analisi, ed è un percorso che permette di trasformare schemi non normalizzati in schemi che soddisfano una determinata forma normale.

In tale processo ad esempio si sostituiscono le relazioni che non soddisfano la BCNF con il risultato di una decomposizione sulla base delle dipendenze funzionali di tali relazioni in altre relazioni che soddisfano la BCNF.

Primo passo semplice di normalizzazione, non sempre valido ed applicabile:

- per ogni relazione che contiene una dipendenza funzionale $X \rightarrow Y$ che viola la BCNF si definisce una nuova relazione XY e si eliminano dalla relazione originaria gli attributi Y

Decomposizioni senza perdite (*lossless*)

Non sempre scomponendo una relazione per ottenere un modello BCNF si ottiene un insieme di dati che preserva tutte e sole le informazioni del modello precedente
Scomponendo la relazione si creano delle nuove relazioni che riaggregate introducono dei dati precedentemente non presenti.

•Una relazione r si decompone senza perdita su X1 e X2 se il join delle proiezioni di r su X1 e X2 è uguale a r stessa (cioè non contiene ennuple spurie)

•La decomposizione senza perdita è garantita se gli attributi comuni contengono una chiave per almeno una delle relazioni decomposte

Uno schema R(X) si decompone senza perdita negli schemi R1(X1) eR2(X2) se, per ogni istanza legale r su R(X), il join naturale delle proiezioni di r su X1 e X2 è uguale a r stessa.

Mantenimento delle dipendenze

•Una decomposizione conserva le dipendenze se ciascuna delle dipendenze funzionali dello schema originario coinvolge attributi che compaiono tutti insieme in uno degli schemi decomposti

Decomposizione in 3NF

•si crea una relazione per ogni gruppo di attributi coinvolti in una dipendenza funzionale
•si verifica che alla fine una relazione contenga una chiave della relazione originaria

- 1NF: Una colonna un valore. Rimuovere gruppi ripetuti
- 2NF: Spezzare in tabelle che descrivano entità separate. Spezzare le PK composte
- 3FN: Rimuovere colonne calcolate e creare tabelle di lookup

Denormalizzazione

La normalizzazione non va intesa come un obbligo, in quanto in alcune situazioni le anomalie che si riscontrano in schemi non normalizzati sono un male minore rispetto alla situazione che si viene a creare normalizzando

In particolare:

- Normalizzare elimina le anomalie, ma può appesantire l'esecuzione di certe operazioni
- La frequenza delle modifiche incide su qual è la scelta più opportuna (relazioni "quasi statiche" danno meno problemi se non normalizzate)
- La ridondanza presente in relazioni non normalizzate va quantificata, per capire quanto incida sull'occupazione di memoria, e sui costi da pagare quando le repliche di una stessa informazione devono essere aggiornate

A volte si decide di non normalizzare o addirittura di denormalizzare strutture normalizzate.

Quando Denormalizzare

- Performance
- Quando gli utenti lo richiedono (anche se questo puo' essere evitato)

Se Denormalizzate

- **Farlo deliberatamente**
- **Avere una ottima regione per farlo**
- **Essere ben consci di cosa questo comporti in termini di performance**
- **Documentare la decisione**

Sistemi Informativi

Storia dei sistemi informativi

Year	Main activities	Skills required
1970s	Mainframe computers were used Computers and data were centralized Systems were tied to a few business functions: payroll, inventory, billing Main focus was to automate existing processes	Programming in COBOL
1980s	PCs and LANs are installed Departments set up own computer systems End-user computing with Word Processors and Spreadsheets makes departments less dependent on the IT department Main focus is automating existing processes	PC support, basic networking
1990s	Wide Area Networks (WANs) become corporate standards Senior management looks for system integration and data integration. No more stand-alone systems. Main focus is central control and corporate learning	Network support, systems integration, database administration
2000s	Wide Area Networks expand via the Internet to include global enterprises and business partners – supply chain and distribution Senior management looks for data sharing across systems. Main focus is efficiencies and speed in inventory, manufacturing, distribution	Network support, systems integration

Catena del valore di Porter

Metodologia di analisi del vantaggio competitivo introdotta nella metà degli anni 80.

E' uno strumento per valutare dinamicamente se e quanto il vantaggio competitivo venga raggiunto, mantenuto e difeso. Può essere utilizzata anche per considerare in maniera formalizzata le opportunità offerte dalle tecnologie dell'informazione.

La *catena del valore* permette, infatti, di considerare l'impresa come un sistema di attività generatrici del valore, inteso come il prezzo che il consumatore è disposto a pagare per il prodotto che soddisfa pienamente i propri bisogni.

Le tecnologie dell'informazione possono influenzare tali attività notevolmente, alcune volte migliorandone semplicemente l'efficacia, altre modificandole profondamente.

Le attività aziendali, per poter valutare la capacità competitiva di un'azienda, sono suddivise in nove categorie generali: cinque sono denominate attività dirette o primarie, quattro attività di supporto.

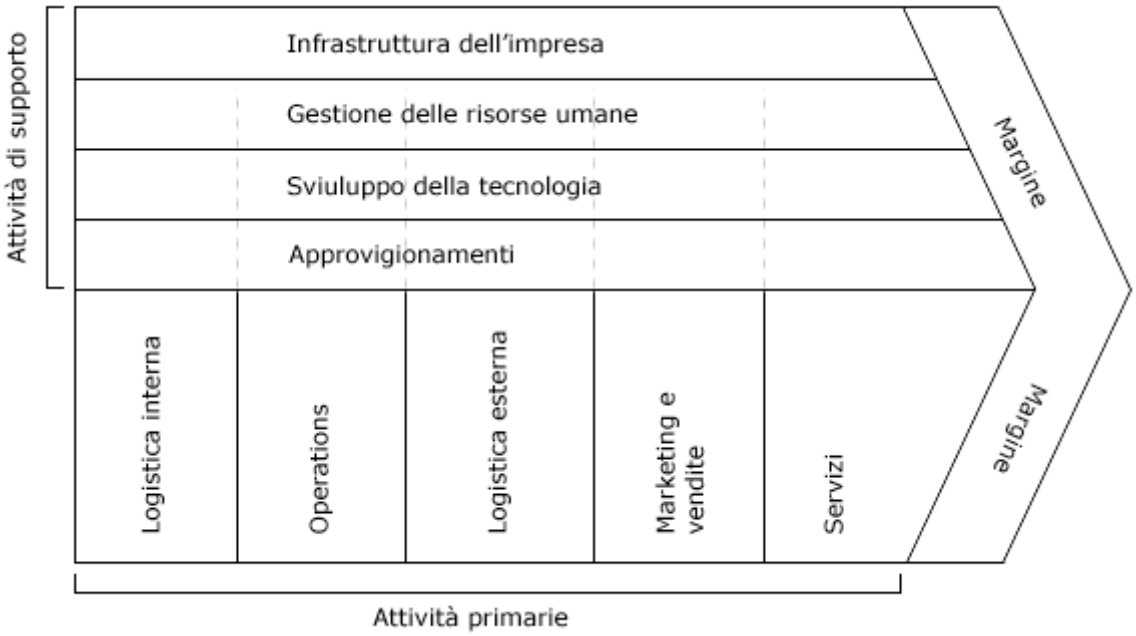
Attività Primarie

- logistica in entrata (beni che "entrano" nell'azienda)
- attività operative (produzione di beni e servizi)
- logistica in uscita (beni che "escono" dall'azienda)
- marketing e vendite
- servizi post-vendita (assistenza tecnico-commerciale, etc.)

Attività di Supporto

Le attività di supporto vengono dette ausiliarie in quanto sostengono le attività primarie e forniscono funzioni estensibili a tutta l'azienda. Si dividono in quattro categorie:

- **approvvigionamento** (riguarda la problematica centrata sull'acquisto delle risorse fisiche impiegate nella catena del valore)
- **sviluppo della tecnologia** (si articola in una gamma di attività finalizzata al miglioramento del prodotto/processo)
- **gestione delle risorse umane** (ricerca, selezione, assunzione, addestramento, formazione, aggiornamento, sviluppo, mobilità, retribuzione,sistemi premianti, negoziazione sindacale e contrattuale, etc.)
- **attività infrastrutturali** (le altre attività quali: pianificazione, contabilità, finanza, organizzazione, informatica, affari legali, direzione generale, etc.)



All'interno della struttura aziendale i sistemi informativi possono essere suddivisi in due macro categorie a seconda della funzione del sistema stesso:

- sistemi Operazionali
- sistemi Informazionali (o direzionali)

Prima di seguire la strada dell'informatizzazione ogni azienda deve essere consapevole che: bisogna investire prima in strategia, organizzazione e comunicazione, solo dopo nella tecnologia.

La scelta del software non ha alcun effetto sulla probabilità di successo se vi è una strategia, un'organizzazione ed una gestione disastrata. Ciò non implica che i software siano tutti uguali, ma significa solo che nessun software porterà al successo un progetto sbagliato.

Ogni sistema dev'essere quindi valutato su ROI o sui vantaggi competitivi che induce nell'organizzazione.

Sistemi Operazionali

Finalità: supportare le attività del segmento operativo di un'azienda

Caratteristiche dell'informazione nei sistemi Operazionali e rappresentazione della realtà:

- informazioni di dettaglio
- in tempo reale
- precise

ERP

Enterprise Resource Planning

Sistemi di base:

- Amministrazione
- Logistica
- Vendite
- Acquisti
- Produzione

Amministrazione

Sistemi di gestione principalmente in ottica contabile, fatturazione, libri contabili, prima nota.

Logistica

Gestione delle movimentazioni dei beni, magazzini, carico e scarico dagli stessi, bolle di viaggio.

Vendite

Ordini, listini, anagrafica clienti, calcolo commissioni ai venditori.

Acquisti

Emissione ordini di acquisto, anagrafica fornitori, listini d'acquisto, centralizzazione dei fabbisogni, ottimizzazione sulle quantità e sui migliori prezzi.

Produzione

Material Reqirements Planning (anagrafica prodotti, distinta base)

Pianificazione della produzione

Considerazioni sugli ERP

- **localizzazione normativa**
- **localizzazione culturale**
- **specializzazione merceologica**

Tipo di processo:

- processo continuo (spaghetti)
- processo discreto (automobili)
- processo a lotto (barche a vela)

Tipo di trasformazione:

- trasformazione predicibile (lamiera -> scatole di latta)
- trasformazione non predicibile quantitativa (maiale -> salame) -> problema dello "sgocciolo"
- trasformazione non predicibile qualitativa (uva -> vino) -> sarà un'annata buona ?

Stagionalità:

- produzione continua (cemento)
- produzione stagionale (marmellata)

Conclusione:

mogli, buoi ed ERP dei paesi tuoi

Esempi di sistemi ERP

Market Share nel 2006 (Gartner)

SAP	28 %
Oracle Suite – JD-Edwards	10 %
The Sage Group (England)	7 %
Microsoft Dynamics (ex Navision)	4 %

Sistemi complementari di un ERP

- MRP – MRP II
- Gestione Impianti (manutenzione)
- Sistemi di gestione finanziaria
- Risorse umane
- Controllo qualità
- Ricerca e sviluppo

MRP

Gli MRP (Materials Requirements Planning) si preoccupano di supportare i processi primari di produzione (attività in linea di produzione, gestione degli approvvigionamenti dei materiali, ecc).

Sono quindi un primo fondamentale sistema complementare ad un ERP, anzi spesso nasce prima dell'ERP col quale verrà poi ad integrarsi.

Una differenza di base tra i due sistemi è che negli MRP i programmi utilizzano tendenzialmente diverse base di dati, mentre gli ERP utilizzano un'unica base condivisa realizzando una più completa integrazione in grado di ridurre tempi e costi di gestione dei dati, di sviluppare una visione d'insieme più completa al fine di supportare meglio i processi decisionali delle funzioni aziendali preposte a questo.

In ogni azienda c'è una persona che deve rispondere alle tre domande:

- Che cosa produrre ed acquistare?
- Quanto produrre ed acquistare?
- Quando produrre ed acquistare?

L'MRP è una tecnica e quindi un sistema informativo che permette di rispondere a queste domande, tenendo conto di importanti vincoli:

- minimizzazione delle scorte
- massimizzazione del livello di servizio
- massimizzazione dell'efficienza del sistema produttivo

Gli MRP-II (Material Reqirements Planning, di seconda generazione) sono la naturale evoluzione dell'MRP attraverso una sua integrazione con l'ERP già in fase di progettazione.

Estensioni verso il mondo esterno

Sistemi Operazionali esterni

- CRM
- e-commerce
- Supply chain management

CRM

Il Customer relationship management ha come obiettivo la fidelizzazione dei clienti.

L'errore più comune è quello di identificare il CRM con un particolare software.

Il CRM non è una semplice sistema informatico, bensì un concetto strettamente legato alla strategia, alla comunicazione, all'integrazione tra i processi aziendali, alle persone ed alla cultura, che pone il cliente al centro dell'organizzazione.

L'attenzione verso il cliente è cruciale e determinante. Per questo motivo il marketing management deve pianificare e implementare apposite strategie per gestire una risorsa così importante, ed avere gli strumenti informatici per gestire questo patrimonio al meglio.

Il CRM si spinge sostanzialmente secondo due direzioni differenti e separate:

- L'acquisizione di nuovi clienti (o "clienti potenziali")
- L'aumento delle relazioni con i clienti più importanti (o "clienti coltivabili") fidelizzando i clienti che hanno maggiori rapporti con l'impresa (definiti "clienti primo piano")

Il sistema informativo di supporto al CRM si articola comunemente in 3 tipologie:

- CRM operativo: automazione dei processi che prevedono il contatto diretto con il cliente.
- CRM analitico: per migliorare la conoscenza del cliente attraverso l'estrazione di dati dal CRM operativo, la loro analisi e lo studio revisionale sui comportamenti dei clienti stessi.
- CRM collaborativo: tecnologie integrate con gli strumenti di comunicazione (telefono, fax, e-mail, ecc.) per gestire il contatto con il cliente.

Le applicazioni CRM servono a tenersi in contatto con la clientela, a memorizzare e gestire le informazioni sui clienti ed a creare modalità d'interagire che possano essere registrate e analizzate.

Possibili strumenti per il CRM:

- chat online;
- forum di discussione;
- risposte alle domande più frequentemente poste dagli utenti (FAQ);
- indirizzi e-mail a cui rivolgersi;
- servizi informativi erogati anche su strumenti ad alta accessibilità (SMS, tecnologia WAP)
- Ticket on-line per la segnalazione di problemi o per la richiesta di assistenza;
- Tracciamento interno di ogni comunicazione "da" e "per" il cliente;
- Archivio dei pagamenti e degli ordini effettuati dal cliente accessibile dal cliente stesso
- etc....

Sistemi locali

Sistemi finalizzati a problematiche tecniche locali

- CAD – Computer Aid **Design**
- CAE – Computer Aid **Engineering**
- CAM – Computer Aid **Manufacturnig**
- Gestione di commessa (Project Management)
- Schedulazione della produzione
- CIM – Computer **Integrated Manufacturnig**

CAM

Alla base del CAM (Computer Aid Manufacturing) c'è il PLC.

Il controllore logico programmabile o programmable logic controller (PLC) è un computer specializzato nella gestione dei processi industriali, che esegue un programma ed elabora i segnali digitali ed analogici provenienti da sensori installati nell'impianto produttivo e ne genera di nuovi diretti agli attuatori presenti sul campo.

CIM

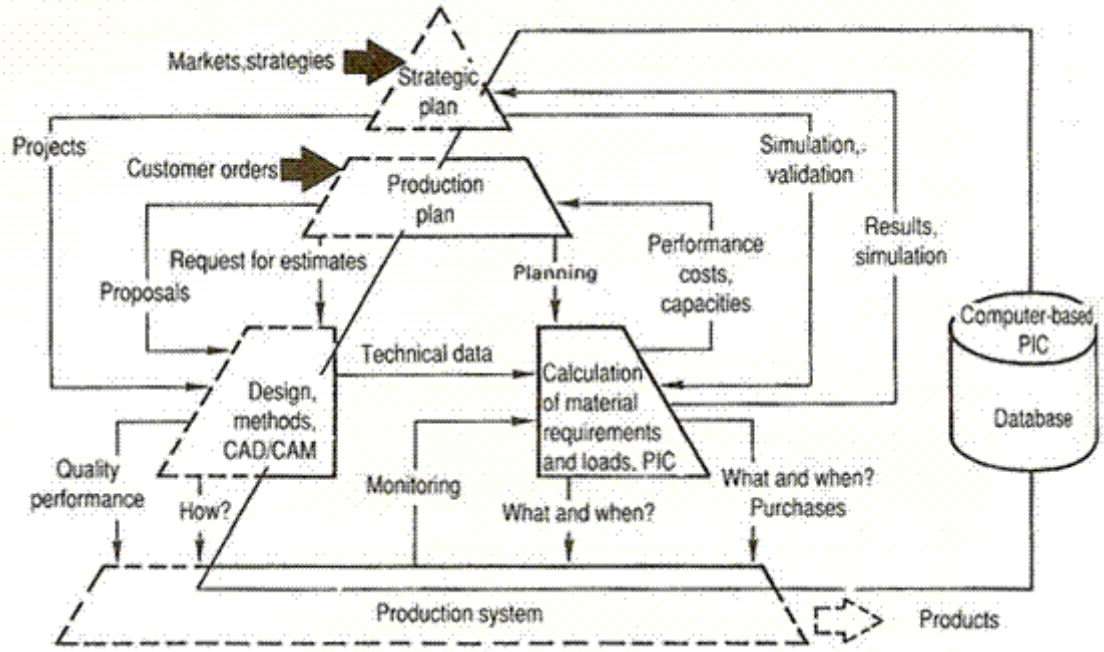
Computer Integrated Manufacturing

La produzione integrata di fabbrica o CIM è l'integrazione tra i vari settori di un sistema di produzione al fine di minimizzare i tempi di sviluppo di un prodotto, ottimizzare la gestione delle risorse ed essere flessibili nella capacità di creare prodotti innovativi.

I vantaggi del CIM sono:

- riduzione delle scorte
- riduzione del time to market
- incremento della qualità del prodotto
- maggiore efficienza della fabbrica

Il CIM si basa su una rete per lo scambio di informazioni che lavora orizzontalmente tra reparti e verticalmente tra settori, in una sorta di struttura piramidale che dal basso verso l'alto vede il processo (sensori e attuatori), il campo (PLC e PMC), la cella (supervisione dei computer industriali) ed infine la gestione (pc per uffici).



Burotica ed automazione del lavoro d'ufficio

Office Automation

- produttività personale (videoscritura, fogli di calcolo)
- gestione documentale (document flow e workflow in senso lato)
- produttività di gruppo (posta elettronica, calendari condivisi, e-conference)

Sistemi Informazionali

Sono detti anche sistemi di Business Intelligence.

Il termine **Business Intelligence**, coniato all'inizio degli anni '90 viene utilizzato per indicare un insieme di strumenti e metodologie per la raccolta e l'analisi dei dati, atti a trasformare i dati stessi in un'informazione utile per gestire i processi decisionali.

Le finalità di un sistema informazionale (o sistemi di Business Intelligence) è di estrarre dalla "materia prima" costituita dai dati contenuti magari in sistemi Operazionali tutte le *informazioni* più o meno nascoste che questi contengono.

Un sistema di BI deve possedere le seguenti caratteristiche:

- ☐ **Facilità d'uso**: presentare i dati in un formato che sia facile da leggere e da interpretare, dove sia facile navigare sui dati stessi seguendo dei percorsi di analisi non necessariamente predefiniti, che faccia un ampio uso di grafici.
- ☐ **Velocità**: possibilità di trattare grandi volumi di dati con tempi di risposta ridotti grazie all'uso di tecniche di modellazione, memorizzazione e indicizzazione dei dati orientate all'analisi invece che all'aggiornamento come accade nei sistemi Operazionali.
- ☐ **Integrazione**: integrare tra loro dati provenienti da fonti differenti, sia interne che esterne all'azienda. Se i dati provenienti dai sistemi operazionali o esterni non sono puliti ed affidabili, prima di entrare nel DW devono subire un processo di pulizia
- ☐ **Storicizzazione**: mantenere la storia di certi attributi selezionati, per permettere analisi storiche mirate.
- ☐ **Identificazione di trend ed anomalie**: identificazione di trend nei dati, confrontando periodi e tipologie diversi. Operazioni possibili con l'utilizzo di strumenti che permettano di effettuare il drill down/drill up (visualizzazione dei dettagli su un certo dato) e di slice & dice (cambiamento delle dimensioni di analisi sui due assi).
- ☐ **Subject orientation**: vedere i dati in modo coerente con i processi aziendali nella loro completezza, attraversando i confini delle singole aree dei sistemi gestionali.
- ☐ **Simulazione scenari**: per alcune tipologie di applicazioni (budgeting, planning) deve essere possibile creare degli scenari di simulazione e confrontarli con dati valori reali
- ☐ **Indipendenza dal reparto I.T.**: data l'enorme variabilità di report ed analisi che sono necessari per un'approfondito studio delle informazioni presenti in un DWH, gli strumenti di analisi e reportistica devono essere in mano agli utenti finali, permettendogli di crearsi da soli i report di cui hanno bisogno
- ☐ **Adattabilità nel tempo**, deve facilmente adattarsi alle variazioni ed evoluzioni dei componenti e delle sorgenti dei dati, oltre che alle sempre diverse esigenze di analisi
- ☐ **Sicurezza**: deve essere possibile controllare in maniera rigorosa e flessibile l'accesso ai dati, che spesso rappresentano informazioni altamente riservate.

Le due filosofie per ottenere informazioni dai dati:

- catalogo e raccolgo (ordino per distillare le informazioni)
- cerco e forse trovo (dal caos accumulato estraggo l'informazione)

Data WareHouse (DWH)

Insieme di dati (tematici, integrati o no, temporali, permanenti) finalizzato al supporto dei processi decisionali. Viene alimentato attraverso processi di importazione e trasformazione dei dati contenuti nei sistemi operazionali ed eventualmente esterni.

Contiene dati di dettaglio o con aggregazioni minime ed è anche detta "base dati di primo livello".

Componenti di un DWH

- modello multidimensionale
- ipercubi
- data mart

Modello Multidimensionale

Si tratta di un'organizzazione delle informazioni realizzata per consentire un efficace utilizzo degli strumenti automatici di analisi.

Può essere realizzata fisicamente su strutture dati proprietarie (database multidimensionali) o su database relazionali attraverso una modellazione dati denominata “star-schema”.

Ipercubi

Data-mart

Insiemi di dati di un DWH focalizzati su di un particolare aspetto del business e resi disponibili attraverso appositi strumenti di analisi e consultazione.

Contiene dati sia di dettaglio che ad un più alto livello di aggregazione, e per questo è anche detta "base dati di secondo livello".

Politiche e tempi di popolazione di un DWH

I DWH possono essere alimentati sia in tempo reale, ma più spesso attraverso trasferimenti periodici dei dati, che vengono travasati all'interno del DWH dopo un opportuna attività di pulizia.

Aree di applicazione di un DWH

- Controllo di gestione
- Logistica
- Controllo di qualità
- CRM
- Gestione del personale

OLAP (On Line Analytical Processing)

Con il termine OLAP si intende la possibilità di effettuare analisi dei dati su strutture multidimensionali in maniera rapida, flessibile ed efficiente, attraverso i servizi forniti da motori di database specifici.

I sistemi OLAP sono la naturale estensione dei DataWareHouse.

Le analisi multidimensionali (in Inglese "Slice & Dice") consistono nel "navigare" i dati lasciando all'utente la facoltà di scegliere interattivamente le informazioni da visualizzare ed i filtri da applicare.

Questa tecnica è detta “pivoting” (in quanto permette di ruotare i dati sui vari assi di riferimento, usando detti assi come perni – pivot), ed è utilizzata in strumenti come Excel sotto il nome di “Tabelle Pivot”.

Un'altra funzionalità dei sistemi OLAP è quella del DRILL, che consente di visualizzare dati a diversi livelli di dettaglio, “navigando” attraverso le gerarchie.

Si parla di drill-up quando l’operazione provoca un’aggregazione delle informazioni, drill-down quando succede il contrario.

La tecnologia OLAP quindi consente all'utente di realizzare da solo le sue analisi, senza la necessità di ricorrere all'aiuto di personale tecnico.

OLTP (On Line Transactiona Processing)

I sistemi OLTP (On Line Transaction Processing) usano un insieme di tecniche software per l'analisi dei dati che a differenza dei sistemi OLAP non prevedono la creazione di banche dati separate.

Le analisi vengono invece effettuate direttamente sui dati di esercizio, e questa soluzione permette di avere i dati sempre aggiornati.

Tuttavia non è applicabile in situazioni dove la quantità di dati da analizzare sia molto elevata.

DataMining

Il processo di mining: dal dato all'informazione

- insieme dei dati
- tipologia delle informazioni da ricercare
- pattern di ricerca
- base di conoscenza
- clustering

Aree di applicazione del Datamining

- analisi finanziaria
- marketing

Knowledge Management

Con questo termine si intende un sistema che presenta la capacità di gestire la conoscenza globale di un organismo (azienda, ente) intesa come somma delle competenze dei singoli, permettendone uno sfruttamento razionale.

La conoscenza aziendale trae origine in genere da tre diverse fonti:

- I dati che risiedono nei documenti, nei database operativi e nei Data Warehouse

- La conoscenza delle persone che lavorano in azienda

- Le fonti esterne (Internet o basi dati di terze parti)

Il K.M. integra queste fonti diverse e le incanala verso una struttura funzionale alle esigenze specifiche dell'azienda in cui viene realizzato.

L'obiettivo fondamentale del K.M è rendere l'organizzazione indipendente dalle singole persone che la costituiscono, “trascrivendo” le conoscenze dei singoli in un sistema organico e centralizzatto.