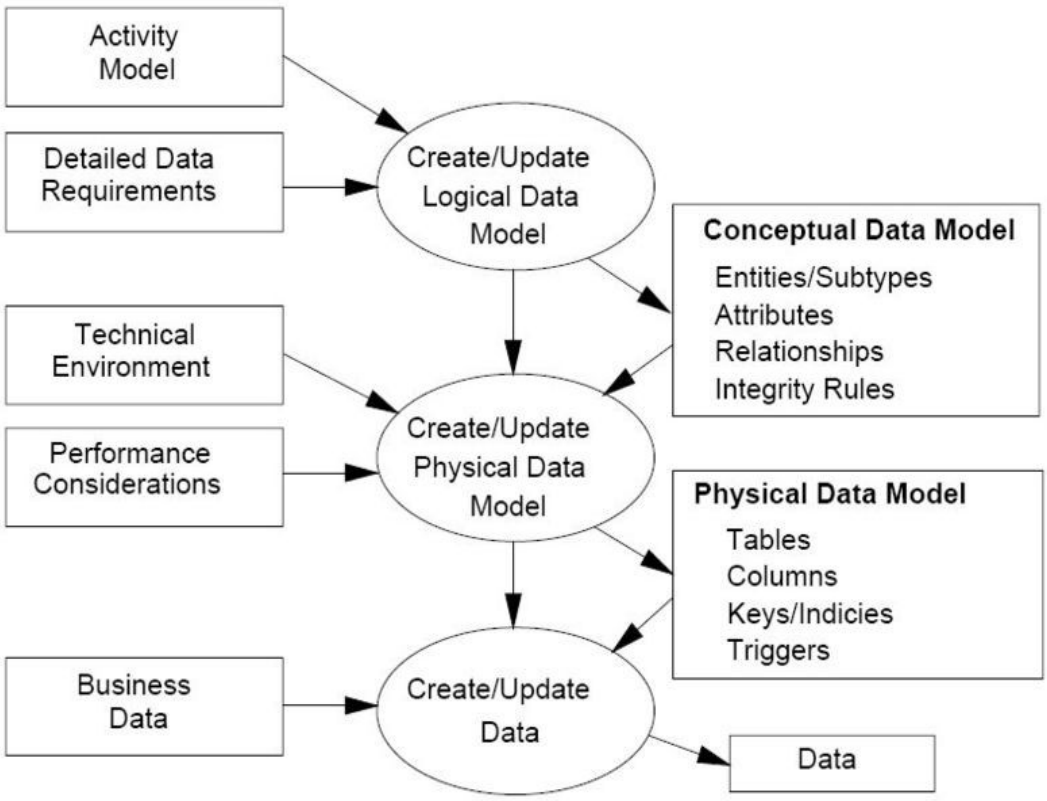
Schema - Model

Levels of database schema

- **Conceptual** schema, a map of concepts and their relationships.
- **Logical** schema, a map of entities and their attributes and relations
- **Physical** schema, a particular implementation of a logical schema



Diversi livelli di ***Data Model***

Cataloghi relazionali

"The schema of a database system is its structure described in a formal language"

Il DBMS mantiene tutte le informazioni sulla base di dati (*metadati*) in delle tavole di sistema che formano il cosiddetto *catalogo*.

Le tabelle del catalogo dovrebbero essere interrogabili attraverso SQL esattamente come le tabelle create dagli utenti.

Esempio (Oracle)

```
select TABLE_NAME
from ALL_TABLES
where OWNER = '<nome-utente>'
```

In alcuni DBMS (Oracle) dato il nome della tabella è possibile ottenere i suoi attributi (nome e tipo) con il comando:

```
describe <nome-tavola>
```

Altri hanno sistemi piu' complessi:

```
SELECT COLNAME
     , COL.TYPENAME
     , COL.LENGTH
     , COL.SCALE
     , CASE
          WHEN REMARKS IS NULL THEN
                       (SELECT COL1.REMARKS
                          FROM SYSCAT.COLUMNS COL1
                          WHERE COL1.TABSCHEMA = COL.TABSCHEMA
                            AND COL1.COLNAME   = COL.COLNAME
                            AND COL1.TABNAME   =
                                     (
                                        SELECT VIE.BNAME AS NOME
                                          FROM SYSCAT.VIEWDEP VIE
                                         WHERE VIE.VIEWSCHEMA = COL.TABSCHEMA
                                           AND VIE.VIEWNAME   = COL.TABNAME
                                     )
                        )
          ELSE REMARKS
        END AS COMMENTO
     , COL.COLNO
  FROM syscat.columns AS COL
 WHERE TABSCHEMA = 'DEPO' AND tabname = 'TAKDSTK'
ORDER BY COL.COLNO
```

Catalogo interno DEFINITION SCHEMA

Catalogo esterno INFORMATION SCHEMA

Indici

“An index makes the query fast”

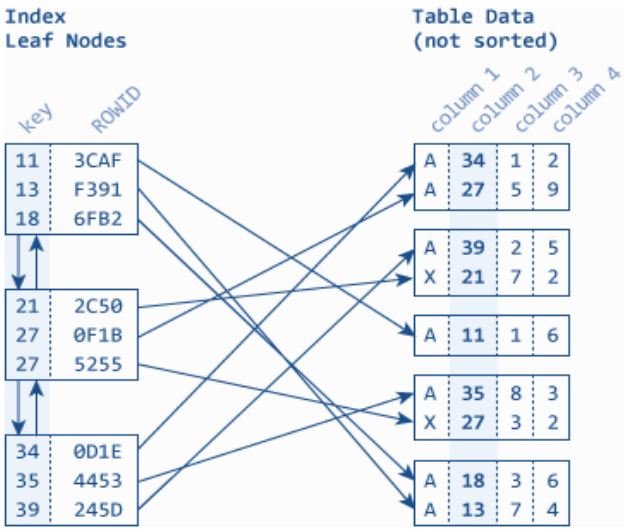
Un **indice** è una struttura dati realizzata per migliorare i tempi di ricerca dei dati.

Se una tabella non ha indici, ogni ricerca obbliga il sistema a leggere tutti i dati presenti in essa. L'indice consente invece di ridurre l'insieme dei dati da leggere per completare la ricerca quando interessano solo una parte dei dati stessi o soprattutto se si effettuano delle selezioni su una parte dei dati.

Ad esempio, se si ha un insieme di dati disordinato, è possibile crearne un "indice" in ordine alfabetico, e sfruttare le proprietà dell'ordine alfabetico per arrivare prima al dato o ai dati cercati applicando una ricerca di tipo binario all'indice ordinato.

Gli indici hanno anche degli effetti negativi in quanto rendono più lente le operazioni di inserimenti e modifica. Quindi prima di definire gli indici occorre valutare quali siano le operazioni più frequenti.

Una errata scelta può comportare un degrado delle prestazioni del sistema dovuto all'overhead introdotto per il mantenimento e l'aggiornamento degli indici, infatti ad ogni operazione di inserimento, aggiornamento e cancellazione si deve intervenire anche sul/sui file indice.



Esempio di indice

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
c – ABORT !!

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 ***ATTENZIONE !!!***

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

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

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

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

ORDER BY - ASC (default) e DESC

DISTINCT (o DISTINCTROW)

COUNT (DISTINCT ....)

SUM, MIN, MAX, AVG, STD

GROUP BY

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

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

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

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

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:
una vista è rappresentata da una query (SELECT), il cui risultato può essere utilizzato come se fosse una tabella

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

Viste materializzate

Alcuni DBMS supportano le viste materializzate (materialized views).
Si tratta di viste che vengono scritte fisicamente su disco e che vengono aggiornate automaticamente a intervalli regolari.

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.

Elemento di base nel DBMS per la **AAA**

AAA stands for "authentication, authorization and accounting"

- autenticazione (authentication)
- controllo degli accessi (authorization)
- tracciamento del consumo delle risorse da parte degli utenti (accounting).

Definito in : "RFC 2903 : Generic AAA Architecture"

Cos'è un RFC (Request for Comments) ?

documenti e standard definiti dall'Internet Engineering Task Force (IETF)

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

Transazioni distribuite: Two Phase Commit (2PC)

Il protocollo di aggiornamento a due fasi (2PC) è un algoritmo distribuito che prevede il coordinamento tra tutti i nodi in un sistema distribuito per convalidare una transazione.

Le due fasi sono :
- **fase di richiesta di validazione**, nella quale il coordinatore centrale (dopo aver aperto ed eseguito la transazione su tutti i nodi) richiedere una convalida da ognuno di essi
- **fase di validazione**, il coordinatore, avendo ricevuto risposta positiva da ognuno dei nodi, completa la transazione confermandola a tutti i nodi coinvolti