DBMS – database management system: software package/system to facilitate the creation and maintenance of a computerized database

Database system: the DBMS software together with the data itself

DBMS>flat file

* Data redundancy/inconsistency reduced
* Isolate the programs from the data
* Provides persistent storage of data between execution of programs/applications
* Multiple users can use and share the data
* More abstract views

Flat file > DBMS

* Cost, Initial set up, Education

Hard disk > main memory

* Cheaper, less volatile, store more data

Main memory > hard disk

* Faster (seek time, rotational delay, block transfer time)

Data Model

* Set of concepts that can describe data (high-level = user perception of data, low-level = how data is stored)

Schema

* Description of a database but not the data itself – abstract data type

Database State

* Data in the database at a moment in time

3 schema architecture

* Internal level (where data reside)
* Conceptual level (how data is stored)
* External level (how data is viewed)

Logical data independence

* Ability to change the conceptual schema without changing the views

Physical data independence

* Ability to change the physical schema without changing the conceptual schema

DBMS language

* DDL = used to define the schema/describe the data
* DML = used to insert, delete and modify the data

ER diagram

* Multivalued attribute = own table (attribute + primary key of entity)
  + Key = combination of attribute + primary key
  + Number of attribute = number of circle 0 multivalued attribute
* Total participation = has to have (child must have a mother)
* Weak entity = defined by entity
* Single line goes to double line when making a table
* Many to many relationship
  + Key of table = primary key of each entity
* Moving entities
  + Single line to double line
  + Single to many
* N-ary relationship where n>2
  + Key of each entity = foreign key on relationship table

Composite attribute vs atomic/simple attribute = bart simpson vs 45

Hierarchical database model

* Record + parent-child relationships
* Only 1:M relationship 🡪 redudancy

Network database model

* Records + Sets
* 1:N relationship, uses circular linked list 🡪 clumsy to use

Relational database model

* each relationship = table

Domain D = set of atomic values (range)

Relation Schema R = relation name and a list of attributes

R = name of the relation

r = actual data

Tuple = row of a relation

Degree of a relation = number of attributes it contains

Cardinality of a relation = number of tuples it contains

Properties of relation

* values of an attribute are from the same domain
* each tuple is distinct

Superkey = no tuples are alike

Candidate key = minimal superkey

Semantic integrity constraints

* state constraints (ex hours > 50)
* transition constraint (ex salaries can only increase)

Trigger

* If then

Assertion

* Must be to do

Hard disk

* Record = fixed length/variable length (Ex. Name)
* File = sequence of records
* Block = storing a number of records in one block on the disk
  + Blocking factor = number of records per block
* Spanned record
  + Records that exceed the size of 1+ block

File organization types

Heap

* Can bulk load data, index key = good
* Search is difficult

Ordered file

* Insert/deletion = harder
* Search = easy

Hash function

* 80% full
* records aren’t in order
* not good if we want to retrieve records in a range
* not good when retrieval is based on an attribute other than the hashed one
  + can use index

Dense

* each data is on index

Sparse

* some data is on index

Primary index

* defined on ordered key data file
* one index entry for each block not each recrod

Clustering index

* defined on ordered, non key data file = not distinct
* one index entry for each distinct value

Secondary index

* provides secondary access for file which already has primary access (can be on any field)
* 1 entry for each record - dense

Multi level index

* index itself is ordered, create an index on the index

RAID

* redundant array of inexpensive disks
* Raid 1 = duplicate, Raid 0 = equally split, Raid 5 = split between two, third = combination of into in first two

NEED TO ADD

* Equations for different organization methods
* Pictures for ER relationships

Relational languages

* Procedural = how to get the data
* Non-procedural = what data is needed

Basic operations

* Selection (sigma) = returns a row
* Projection (pi) = returns columns
* Cartesian product (#R \* #R)
* Union
  + NOTE: same number of attributes
  + Returns all rows in either A or B
* Intersect = in both A and B
* Cartesian product
  + No arm = all rows with same values in overlapping column
  + Arm = all rows in side of row + all rows with same values in overlapping column
* set difference (must have same number of attributes)
* Division
  + A/B = all columns in A not in B with overlapping values