01. Database Basics

[PM Jat, DAIICT, Gandhinagar]

Contents

- What is Database?
- How do we represent databases?
- DBMS based computing architecture
- Database and Database Schema
- Three Schema Architecture

What is Database?

In simple terms, Database is a collection of "Relevant" "Data".

We elaborate terms "Relevant" and "Data" next.

A database does not have a random collection of data; it typically

- Records all facts about some "application" context.
- Records necessary information of all events that occur as "business process", for example an order is received; an items is added in store, and so forth
- Stored such that data manipulation operations can be executed efficiently.

Data

Data is an *atomic* value; represents a fact about some *entity*.

For instance, "Amit Kumar" is value for *attribute* name of a student entity; "28-June-1986" value for attribute date-of-birth of the student, and so forth. A set of values for various attributes describe an *entity*.

Data being atomic mean a "single value" for an *attribute*.

Relevant Data

Databases are built for some purpose. A data being relevant mean, data is required for meeting the objective of building the database.

For example, do we need to record names of dependents of an employee in a company database? Answer would depend if the company provides certain benefits to dependents of employees then it may be required otherwise not required.

Hereby we say that data is "relevant" if it is required in database, otherwise not.

Basically, intuition here is, we are able to draw some boundary to separate out "data of the database" in data of universe.

Book elmasri/navathe uses a term called "mini data world", while the book Korth uses a term "enterprise" for capturing the notion of "boundary" for what is relevant and what is not?

Representation of databases

We have certain data modeling techniques that are used for representing databases. Following are two most popular techniques.

Entity-Relationship Models:

- Database is seen as set of *entities* of different types and interactions between them.
- Figure [XIT-ER] below shows a set of student, program, and department entities; an entity in its set is described by values [data] for its attributes.
- ER model is "Conceptual Model", and primarily used for documentation purpose.

Relational Model (Implementation Model)

- In this technique, Database is represented as a set of *relations*, also called as tables.
- Each relation is set of tuple, or rows.
- Relational model is de-facto standard for implementing enterprise databases.

Example #1 (Entity-Relationship Models)

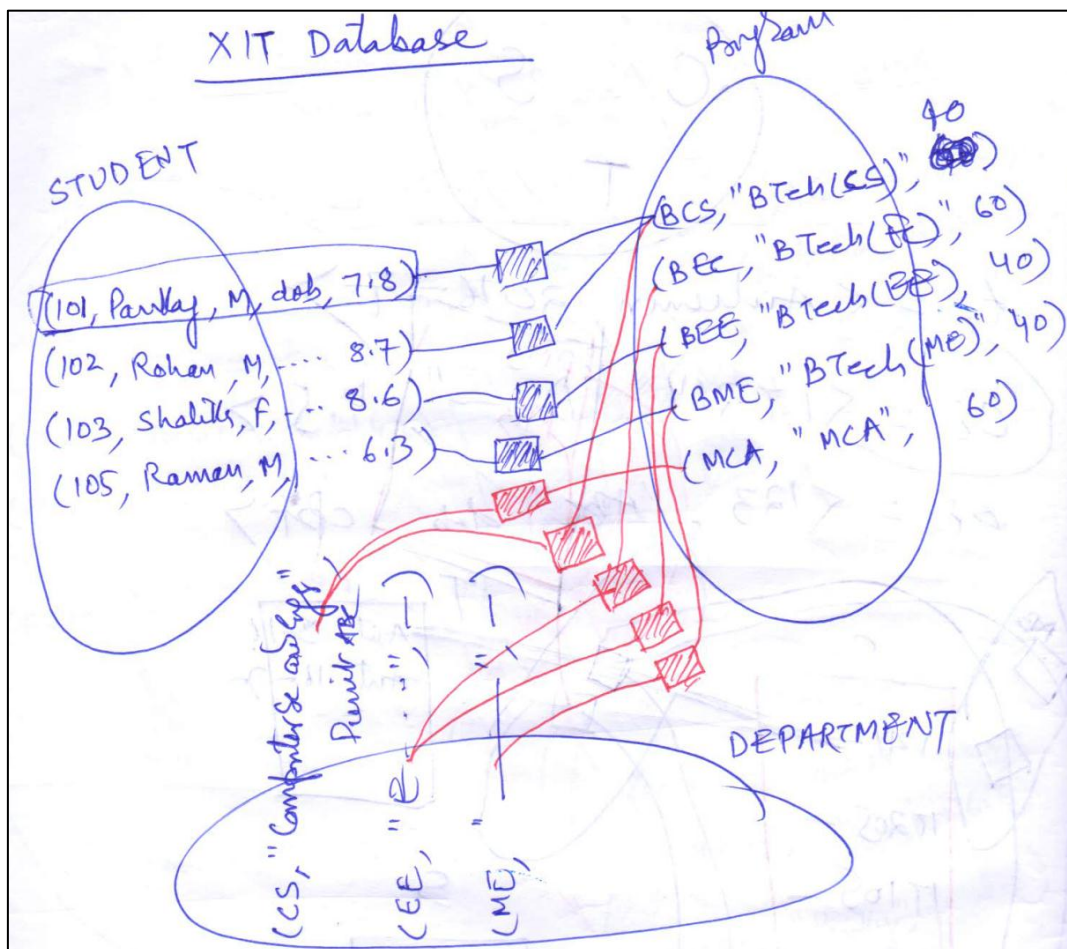Database can be expressed as set of sets of some entity sets and interaction sets.

Figure 1: XIT Database in terms of Entities and their interactions

Where
S = {s1, s2, s3, …} - set of students
P = {p1, p2, p3, …} - set of program
D = {d1, d2, d3, …} - set of departments
SP = set of events/instances of student enrolling in a program
PD = set of events/instances of program getting offered by a department

Example #2 (Entity-Relationship Models)

Figure below partially depicts a database scenario (da-acad, we will be introducing shortly)
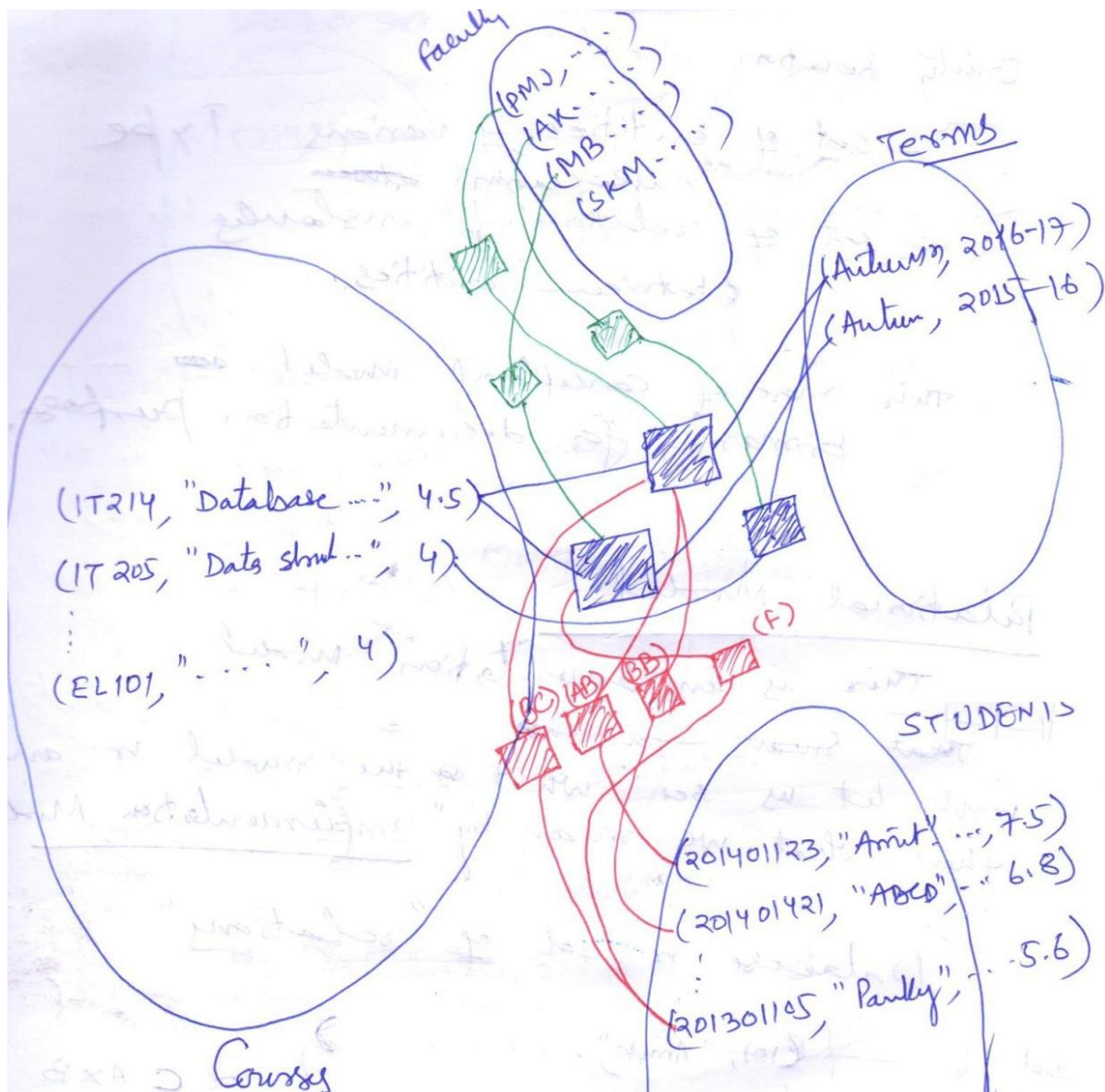

Figure 2: DA-Acad (partial) Database in terms of Entities and their interactions

DA-Acad DB={S, C, T, F, CT, CTF, CTS}

Where

S = {s1, s2, s3, …}; set of students

C = {c1, c2, c3, …}; set of courses

T = {<Autumn,2016>, <Winter,2017>, … }; set of terms

F = set of faculty

CT = Set of instances of Course offerings?

CTF = set of instances of "course offering" getting a faculty associated?

CTS = set of instances of students registering in offerings?

Example (relational representation):

XIT Database – set of relations/tables

DB = {Student, Program, Department}

Where Student, Program, Department are three relations as shown below.

Student

StudentID	Name	ProgID	CPI
101	Rahul	BCS	7.5
102	Vikash	BIT	8.6
103	Shally	BEE	5.4
104	Alka	BIT	6.8
105	Ravi	BCS	6.5

Program

ProgID	ProgName	Intake	DID
BCS	BTech(CS)	40	CS
BIT	BTech(IT)	30	CS
BEE	BTech(EE)	40	EE
BME	BTech(ME)	40	ME

Department

DID	DName
CS	Computer Engineering
EE	Electrical Engineering
ME	Mechanical Engineering

Figure 3: XIT database in terms of Relations

Operations on Databases

Following are main operations on database (also referred as database manipulation operations).

Are of two types:

- Update operations: that changes data of databases; typically add, modify, and delete entities.
  - Add more facts [entities and their interaction]
  - Modify existing data
  - Delete existing facts

  For example - operations on XIT database

  - Add a student (id=234, Name='Aman', ..) in program bcs
  - Update CPI of student having id = 102 to 7.8

- Query: get answer of a query from the database
  - List (ID, Name) of BCS students having CPI > 7.0
  - Give me program-wise student count for all programs
  - How many students are studying in CS department?

It is the business events that trigger the database update. For example following events will be update the corresponding sets in "da-acad" database –

- Add an course - adds an element in set C
- Add new Term - adds an element in set T
- Offer a course in a term - adds an element in set CT
- A faculty is assigned to a course - adds an element in set CTF
- A student register in a course – adds an element in set CTS
- Faculty uploads course grades – modifies attribute values of elements in set CTS (for all students registered in given course in given term)
- Result processing – computes and updates SPI and CPI of each student for current semester!

When data is data?

- For human mind, table below contains some data!
- But is it data for a computer too?
- If stored in Excel, it is more data than doc/html/pdf.
- If stored in any DBMS like PostgreSQL/MySQL/SQLite) as relation, then it is more data than stored in Excel.
- Why is it so?

Machine must be able to "refer" a data item by name; for example "cpi of a student identified by id=1234".

"Machine Processibility" is the key?

This is difficult (for machine) when you have data in text/html/doc/pdf?

When represented in excel, a data item is locatable but not by name but by cell location.

This is why we say that it is less data for machine than we represent in relation form in a dbms like postgresql or mysql.

Point here is, for machines, data is data only when we can identify an data item by name and value.

studentid	name	progid	batch	cpi
200711001	Charu Chawla	11	2007	6.12
200711002	Amit Khanna	11	2007	7.12
200711003	Kamla Kiran	11	2007	7.50
200711004	Raj Kumar	11	2007	4.00
200711005	Raj Tiwari	11	2007	5.56
200811001	Rama Kant	11	2008	8.12
200811002	Akshya Gupta	11	2008	9.22
200811003	Unnati Gupta	11	2008	5.52
200811004	Mridula Singh	11	2008	4.25
200811005	Amit Ajaad	11	2008	6.56

Figure 4: when data is data?

Recap:

- What is database? Database is a collection of "relevant" "data"
- Database Representation.
- (1) Set of *entities* and their *interaction instances*!
- (2) Set of relations
- Few examples
- When data is data?

Database Characteristics

A modern database typically has following characteristics-

- Persistent
- Integrated and Shared
- Self describing through meta-data
- Authorized Access
- Simple interface to update (add/modify/delete) and query the database in logical manner – SQL like query language.
- Accessed through a Database Management System that efficiently enables said characteristics.

Persistent: database is stored on secondary storage.

Integrated and shared: data of different users are stored in a single database and shared by all concerned users.

Self describing nature of Databases: database definition is not defined in *applications* (programs); database itself contains its "structural" definition as a part of database and more importantly in the form of data. [Structural Information is stored as part of Database and not in Application]

Authorized Access: Database is owned by some user; other users may be granted permission for viewing or updating. It is also possible to grant partial access to a database, a user is allowed to view a subset of a database.

Simple interface to update (add/modify/delete) and query **the database**: Though databases are actually used on disk files, and often have complex file organization. User should not be required to deal with such files; rather use should be able to view and manipulate databases as some logical units like "entities" or "relations". For working with databases, we have query language like SQL.

Database Constraints

Database constrains are basically "data existential rules".

"Rules" that must hold true on data (values) in a database.

For example in DA-ACAD database

- StudentID is key attribute of student entity; that means no two student entities can have same value for this attribute.
- A course offering necessarily need to have an instructor associated with; and exactly one instructor.
- An elective can be taken only from respective domain, and so forth

Any data that violate such rules cannot exist in database; if do, then database is said to be in invalid state; such a state is called as "**inconsistent state**" of database.

The interpretation of "Inconsistent", here is database is not consistent with its constraints (rules)

Constraints are part of "database description", referred as *database schema*, and any valid state of database should satisfy these rules.

Database Instance, state, and Schema

The term "database" that we have been learning so far, refers "database instance". All sketches seen so far depict *database instance*.

A snapshot of a database instance (values) is referred as *database state*. In any real system, databases keep updating; and continuously changing its state.

We have another concept "database schema".

Database Schema-

While database instance actually holds data; **database schema** describes "structures of database". Database structure, implicitly, also includes description of *database constraints*.

In other words, database schema contains description of database structure and its constraints.

- Structure describes
    - What entities? Or what relations if represented in relational model.
    - What "attribute values" for each entity (or relations)

- Constraints
    - Defines what is "domain" for each attribute of an entity (or relation)?
    - What is key attribute?
    - What interaction it has with other entities?
      is that interaction is mandatory?;
    - What is cardinality of interaction?
      That is, to how many entities "an entity of a type" might be interacting with; how many other entities, an entity can associate with; and so forth.

Below are depiction of XIT database schema in "entity relation-ship representation" and relational representation-

entity relation-ship representation	relational representation
Figure 5: Schema of XIT database in ER Model	Figure 6: Schema of XIT database in Relational Model
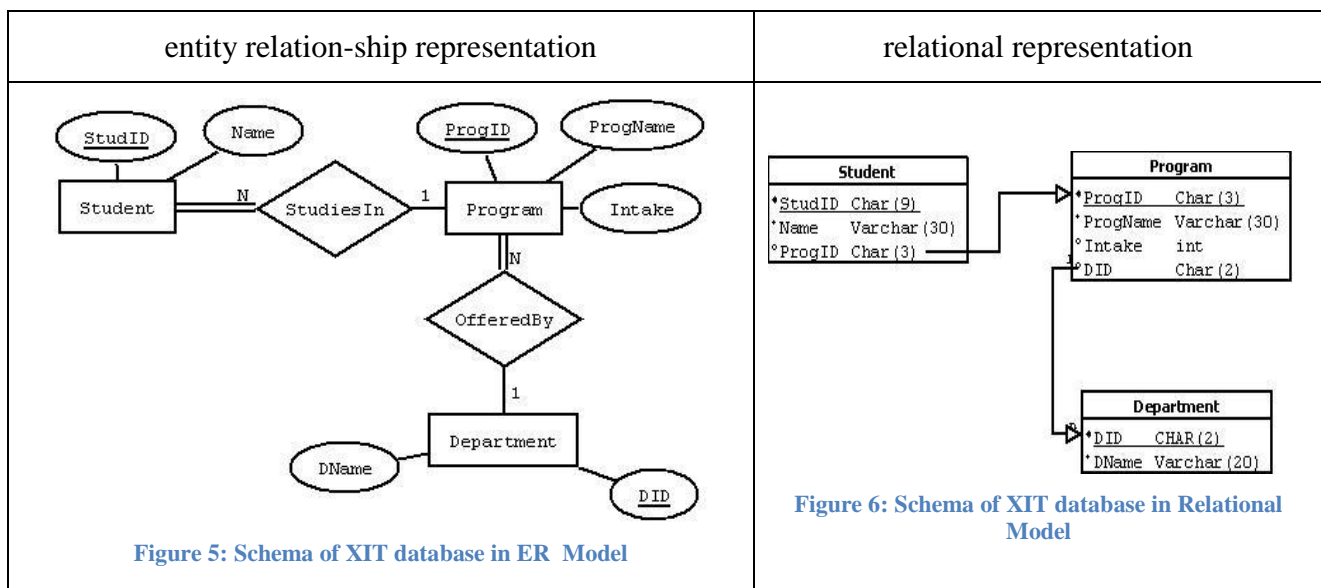
Also below is, XIT schema in SQL-DDL

```
CREATE TABLE department (
    DID CHAR(2) PRIMARY KEY,
    DNAME VARCHAR(30) NOT NULL
);
CREATE TABLE program (
    PID CHAR(3),
    PNAME VARCHAR(20) NOT NULL,
    INTAKE SMALLINT,
    DID CHAR(2) REFERENCES department(did)
        ON DELETE SET NULL ON UPDATE CASCADE,
    PRIMARY KEY (PID)
);
CREATE TABLE student (
    StudID CHAR(3),
    Name VARCHAR(20) NOT NULL,
    ProgID CHAR(3) REFERENCES program(pid)
        ON DELETE CASCADE ON UPDATE CASCADE,
    cpi decimal(4,2)
);
```

Figure 7: XIT schema in SQL-DDL

Here we have three different description of database schema – ER, Relational, and SQL-DDL. Each description serves some specific purpose.

- ER description is often used for documentation purpose.

- Relation description is used for documenting "implementation structure of database".

- Schema in SQL DDL script, as a program, becomes input to a database management system so that **empty database instance** is created on a computer system.

DBMS based Computing Architecture

There are certain issues while we perform database operations. First we take note of issues, and then see how DBMS based computing architecture addresses these issues.

Issues/Concerns in performing database operations:

- Data are stored in disk file. Maintaining efficiently searchable data file is complex task. Suppose, B+ tree is an efficient searching mechanism, implementing this on secondary storage is complex. We would want this complex organization to be transparent from applications.
- "data manipulation" functionality, for example search based on key or some other attribute, insert an entity, modify, delete, or so, is almost repeatable in every database application. We would not want this to be part of application; instead want external and simply reused in every application.
- Application's data access code is dependent on underlying file organization. As a result we cannot independently improve upon file organization. Require changing application code as well.
- Issues related to concurrent access by multiple users, authorized access, dealing with system failures, or so

DBMS Approach:

A DBMS based Computing Architecture was suggested to deal with said issues; figure below (from elmasri/navathe) depicts the architecture. In this approach DBMS takes charge of performing all manipulation operations while providing simple interface to the users.

When user submits database operation to DBMS using interface language like SQL; it in turn translates the request into file manipulation system calls or instructions, after checking syntax etc.

DBMS stores data on disk files; also keeps schema information of every database in its "dictionary" as meta-data.

DBMS maintains sophisticated file organization and indexes for accessing data in data file. Also DBMS implements complex set of algorithms to perform various user operations.

DBMS Definition:

Book Elmasri/Navathe gives following definition of

DBMS-

Figure 8: DBMS based Computing Architecture

*"DBMS is general purpose software system that
facilitates the processes of defining, constructing, manipulating, and sharing the databases among
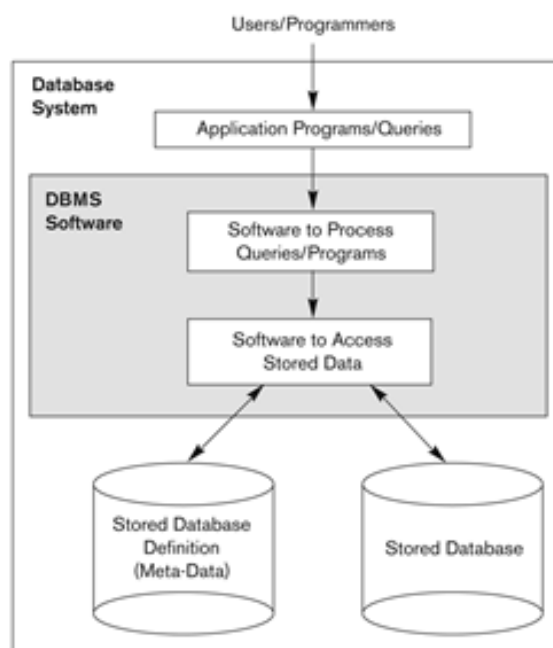various users and applications"*

DBMS is a complex system; a figure below (Fig 11: DBMS-COMPONENTS) from book Elmasri/Navathe depicts its various components and their roles.

DBMS (Database Management System) provides following functionality

- Persistent storage with "Data Abstraction": Database is represented (and manipulated) through some logical representation
- Data manipulation interface – through a query language.
- Ensures "database integrity". DBMS would reject a manipulation operation on a database that violates database constraint.
- Transaction Management: safe shared concurrent access by multiple users and recovers from system failures.
- Database Security: Authorization based accesses

Downside of using DBMS?

- Requires more resources.
- Due to a translation and computing layer in between, performance gets down
- Deployment of application becomes complex.
- Due to these reasons, many popular proprietary applications do not (did not) use DBMSes; reasons could be: performance, cost, proprietary, compact, etc.
- About a decade back mobile or other devices could not afford to use DBMS due to limited resources. Today you have SQLite and similar product, though!
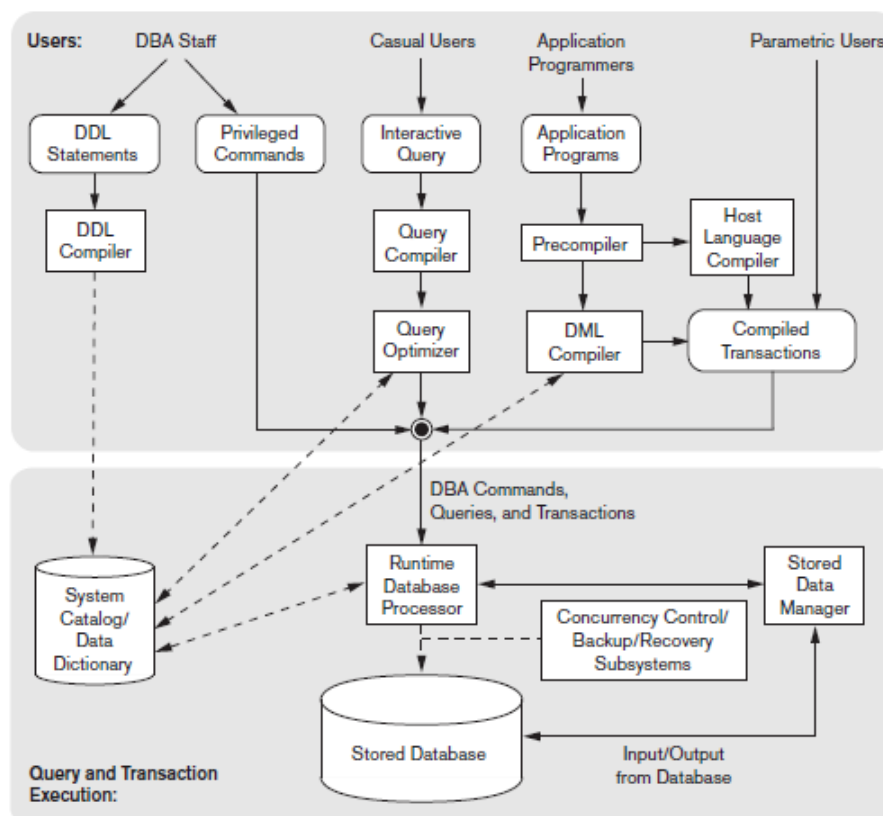

Figure 9: DBMS Components

Meta-Data: Database Schema as Data

One of the characteristics of database is self-describing. DBMS stored schema information of every database in its "catalog" or data dictionary.

Schema Information as data is what we refer as Meta-Data, that is, ==Information about database data in the form of data==? Given below is snapshot of queried schema information from postgresql catalogue. Query used to fetch this information is given in footnote [1]

table_name character varying	column_name character varying	data_type character varying
department	did	character
department	dname	character varying
student	studid	character
student	name	character varying
student	progid	character
student	cpi	numeric
program	pid	character
program	pname	character varying
program	intake	smallint
program	did	character

Figure 10: Database Metadata

---

[1] SELECT table_name, column_name, data_type FROM information_schema.columns
WHERE table_name in (select tablename from pg_tables where schemaname='xit')
and table_schema = 'xit';

Data Abstraction

What is Data Abstraction in general? Consider example below, hopefully illustrates the notion? Observe how floating points are internally represented, and relevant operations are actually performed. User is transparent to underlying binary representation, and procedure of performing floating point operations.

There are two kind of transparency we have in "data abstraction"

- Representation Transparency (how data re represented)
- Operational Transparency (how operations are performed - procedure)

Programming languages do provide such *typed abstraction* over binary representation and manipulation of data.

What we work with	Internal Representation (Float (IEEE754 Single precision 32-bit)
A=134.0625	$134.0625 = 1.00001100001 \times 2^7$ $0x43061000 = 01000011\ 00000110\ 00010000\ 00000000$ Sign Exponent Mantissa 0 10000110 00001100001000000000000
B=-2.25	-1.001×2^1 $0xC0100000 = 11000000\ 00010000\ 00000000\ 00000000$ Sign Exponent Mantissa 1 10000000 00100000000000000000000
A x B	Multiplication of Mantissa, Addition of exponent, and so

Figure 11: Programing Languages provide typed abstraction over binary representation of floating point numbers

Data Abstraction by DBMS

Same notion of data abstraction is applied on relations. DBMS provides -

- Representational Transparency: allows representing database in some logical view while hiding how data are stored in disks on file system.
- Operational Transparency: enables performing operations on database without letting us delved in underlying complex algorithms.

For example, Relational is a popular database representation and manipulation model. RDBMS is DBMS based on Relational model, and provides representational and operational transparencies.

- Defining and manipulating database as relations or table while data are actually stored on disks.
- Perform various manipulation operations on relations in logical manner (rather than how actually they are performed by RDBMS). Obviously DBMS uses sophisticated algorithms for performing database manipulation operations (add/modify/delete, and query/search)
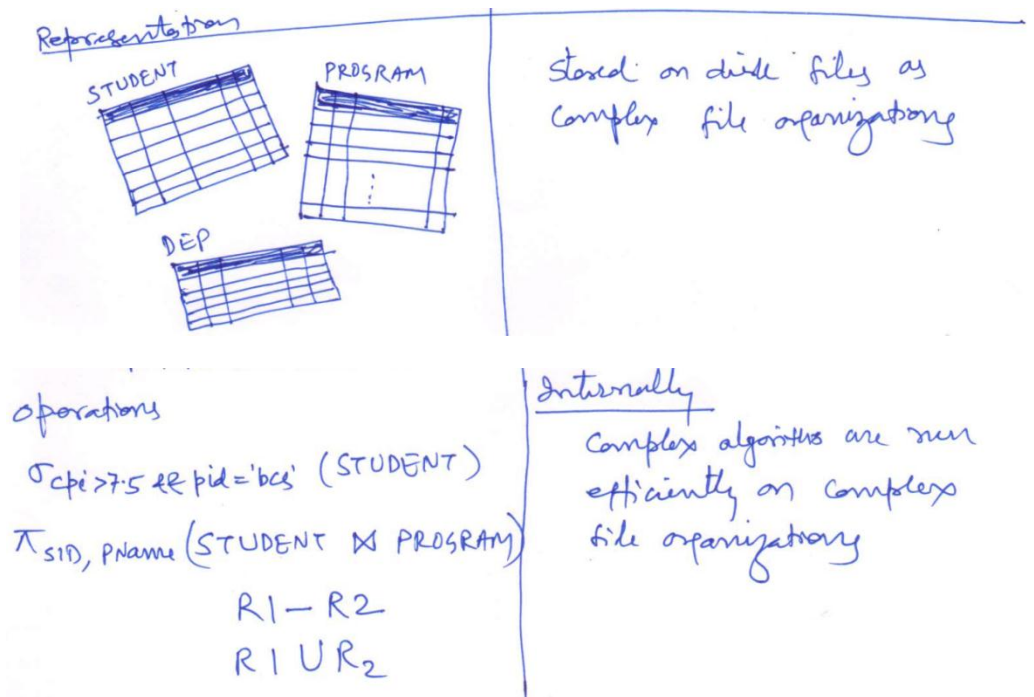
Figure 12: Relational abstraction

RDBMS provide SQL support for manipulating databases. Database is manipulated as tables, rather than data structure on disk files.

This way RDBMS hides complexity of performing operations. Below are some SQL statements for certain database operations-

INSERT INTO STUDENT VALUE('201001123', 'Ankit', ...);

UPDATE REGISTERS SET GRADE = 'AB' WHERE STUDENT_ID = '201001123'
    AND COURSE_NO = 'IT321' AND ACAD_YEAR = 2012;

SELECT * FROM STUDENTS WHERE CPI >=8.0 AND BATCH = 2009;

Actual algorithms that perform these operations are quiet complex. They work on disk files corresponding to concerned relations.

Three Schema Architecture

American National Standards Institute (ANSI) and Standards Planning and Requirements Committee (SPARC) identify three levels of defining database schema (structures).

Diagram below from book elmasri/navathe depicts the architecture.
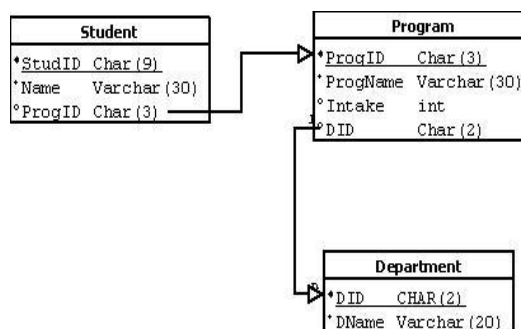

Figure 2.2 The three-schema architecture.
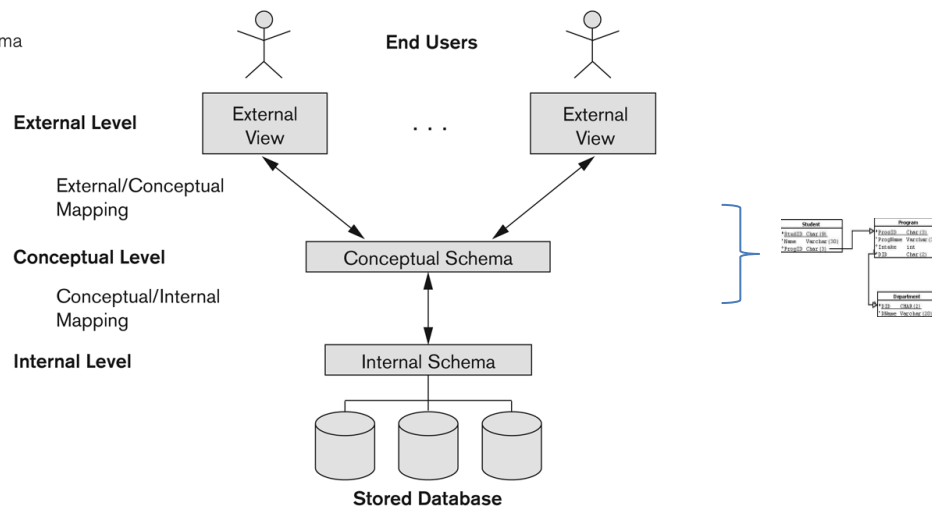
We have three layers of schema.

Middle Layer - Conceptual Schema.

> Here structure of database is defined in some logical manner. For instance database is represented as set of "Tables". Each Table has its Schema (and constraints). Therefore we have some schema here. For example as following for XIT Schema-

Lowest one is "Physical/Internal Schema"

- Data are stored physically stored on disk files. No data at conceptual level.
- Database operations are executed by scanning disk files.
- Data are stored in Complex File Structures and often searches are enhanced by providing access path like B+-Tree, hashing, or so.
- Internal schema primarily means "File organization" and "Data Access paths". File organization mean layout of data records on disk file. File organization also include clustering of data in some order and in some partitions.
- B+ tree and Hashing based physical indexes are most popular searching technique.

External Schema or User schema:

- Different users might see different subset of database, and see a schema defined in independent technique.
- Few examples:
  - One such example is external user might see relational view of some subset of legacy system
  - A user might see object view of database while having relational implementation
- Other use of external schema could be hiding actual database schema.

Why three schema architecture?

What could have been motivating factors of three schema architecture?

1. Abstraction is desirable
   - Data Manipulation on disk files is complex.
   - Abstraction has been effective technique to deal with complexities
   - Following is abstract view of XIT database
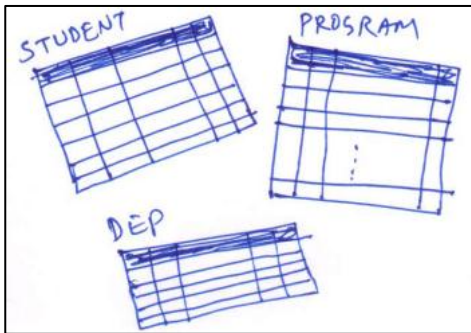
==> Schema for such abstract view is called "Conceptual Schema"

2. Efficient execution of operations is also important
   - Efficient File organization
   - Access Path – disk based B+ tree is most popularly used.

==> Physical/Internal Schema

Now this arrangement gives "Data Independence" at lower level. That is lower level data/file structure (schema) can be modified (for improvement) without affecting high level schema.

For instance, at physical level, algorithms used for performing data operations are dependent on file organization. To improve the efficiency of execution of data operations, it may be required to change the file organization and access paths. Data independency enables that.

How independence is accomplished?

Often database operations are initiated by user at external schema level; whereas actual execution happens at internal schema level (data are actually stored at physical level)

A data reference at external schema should map to data at physical level. Therefore DBMS maintains "mappings" from higher level schema to lower level schema, and from lower level to next level upto the data on the disk.

This mapping actually helps in accomplishing "data independence".

If any change in schema at lower level happens, appropriate changes are made in mapping!

Downside of three schema architecture?

To reach a data item referred by user at external level to corresponding data at physical level requires two stage resolutions through mappings.

This may become too much of computational overhead and takes in executing a database operation (query).

Also, there would be some data representational transformations are happening for appropriate schema model at respective level.

Due to these reasons, most DBMS, RDBMS at least, do not implement full three-schema architecture, and typically do not separate schema at external level.

You can emulate though by creating wrappers on top it. For instance there are lots of Object Relational Mapping tools are used in the industry, and act as bridge between relations and *objects*.