UNIVERSITAS INDONESIA

FAKULTAS ILMU KOMPUTER

2

Database Concept and Architecture

CSF2600700 - BASIS DATA

Acknowledgements

This slide is a modification to supplementary slide of
“Database System”, 6th edition, Elmasri/Navathe, 2011: **Chapter 1 Introduction to Databases**
and “I**ntroduction to Databases**” used in “Basis Data” course in academic years 2018/2019 in the Faculty of Computer Science, Universitas Indonesia

Outline

1. Data Models

- Categories of Data Models

- History of Data Models

2. Schema

- Three-Schema Architecture

3. DBMS Component

4. DBMS Architecture

Data Models

Data Models

A set of concepts to describe the **structure** of a database, and certain **constraints** that the database should obey.

Data Model Operation

Operations for **specifying database retrievals and updates** by referring to the concepts of the data model.

Operations on the data model may include basic operations and user-defined operations.

Categories of data models

Conceptual (high-level, semantic)

Data models: Provide concepts that are **close to the way many users perceive data**. Such as: entity, attribute, relationship among entities (will explain more detail in ER model)

Physical (low-level, internal)

Data models: Provide concepts that describe details of how data is stored in the computer. Ex. Tree, Graph, dsb

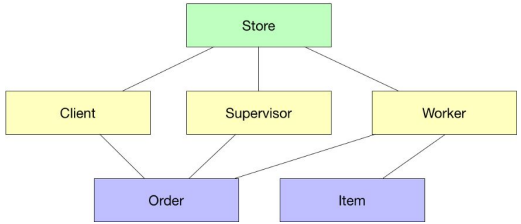
Implementation (representational)

Data models: Provide concepts that fall between the above two, balancing user views with some computer storage details. Such as: relational, network or hierarchical data model

History of data models

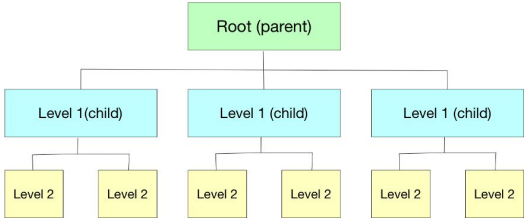
Network Model

- ➔ The first one to be implemented by Honeywell in 1964-65 (IDS System).
- ➔ Adopted heavily due to the support by CODASYL (CODASYL - DBTG report of 1971).
- ➔ Later implemented in a large variety of systems - IDMS (Cullinet - now CA), DMS 1100 (Unisys), IMAGE (H.P.), VAX -DBMS (Digital Equipment Corp.).
- ➔ Data in a **Network** in terms of Interdependencies and Connections Among Data Items
- ➔ **Graphs**



Hierarchical Data Model

- ➔ Implemented in a joint effort by IBM and North American Rockwell around 1965.
- ➔ Resulted in the IMS family of systems. The most popular model. Other system based on this model: System 2k (SAS inc.)
- ➔ Data in **Hierarchies** in terms of Interdependencies and Connections Among Data Items
- ➔ **Tree**

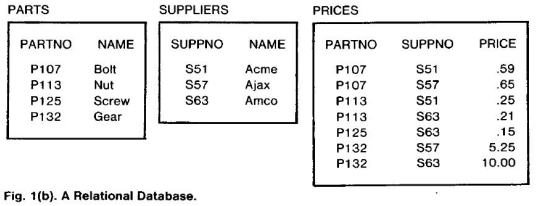


images: https://medium.com/@rpolding/databases-evolution-and-change-29b8abe9df3e

History of data models (Continued)

Relational Model

- Proposed in 1970 by E.F. Codd (IBM),
- First commercial system in 1981-82.
- Now in several commercial products (DB2, ORACLE, SQL Server, SYBASE, INFORMIX).



PARTS

PARTNO	NAME
P107	Bolt
P113	Nut
P125	Screw
P132	Gear

SUPPLIERS

SUPPNO	NAME
S51	Acme
S57	Ajax
S63	Amco

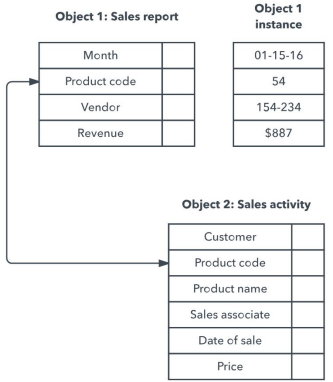
PRICES

PARTNO	SUPPNO	PRICE
P107	S51	.59
P107	S57	.65
P113	S51	.25
P113	S63	.21
P125	S63	.15
P132	S57	5.25
P132	S63	10.00

Fig. 1(b). A Relational Database.

Object-Oriented Data Model(s)

- Several models have been proposed for implementing in a database system.
- One set comprises models of persistent O-O Programming Languages such as C++ (e.g., in OBJECTSTORE or VERSANT), and Smalltalk (e.g., in GEMSTONE).
- Additionally, systems like O2, ORION (at MCC - then ITASCA), IRIS (at H.P.- used in Open OODB).

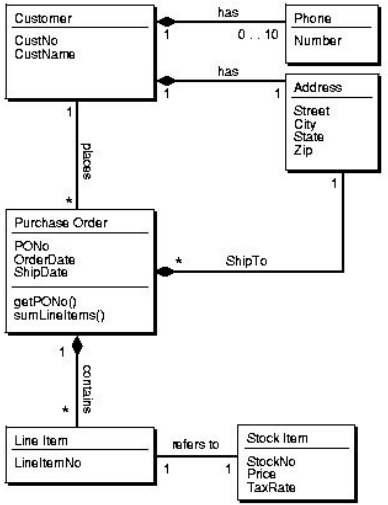


images: https://medium.com/launch-school/a-brief-history-of-postgresql-36d8d392c611
https://www.oreilly.com/library/view/hands-on-big-data/9781788620901/b2b76205-d932-4afa-957f-7013376c553e.xhtml

History of data models (Continued)

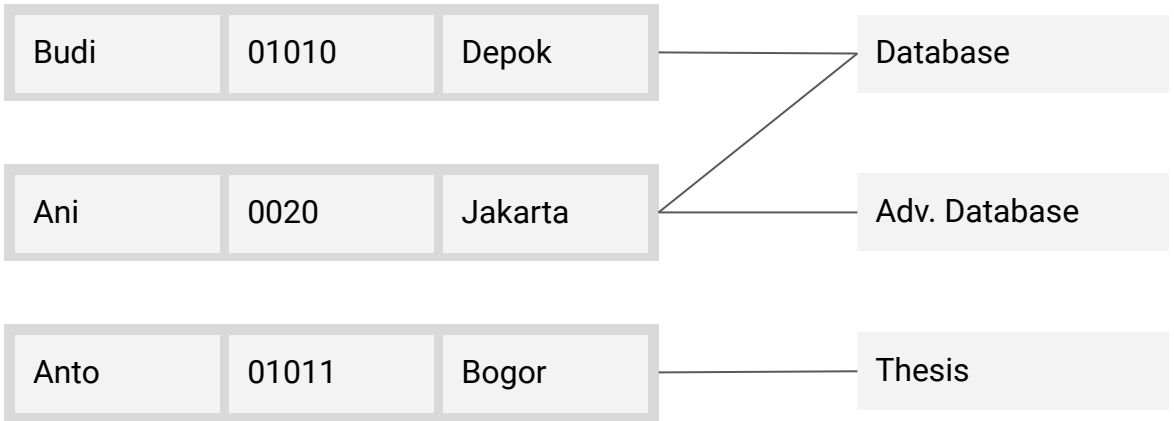
Object Relational Model

- ➔ Most Recent Trend.
- ➔ Started with Informix Universal Server.
- ➔ Exemplified in the latest versions of Oracle-10g, DB2, and SQL Server etc. systems.

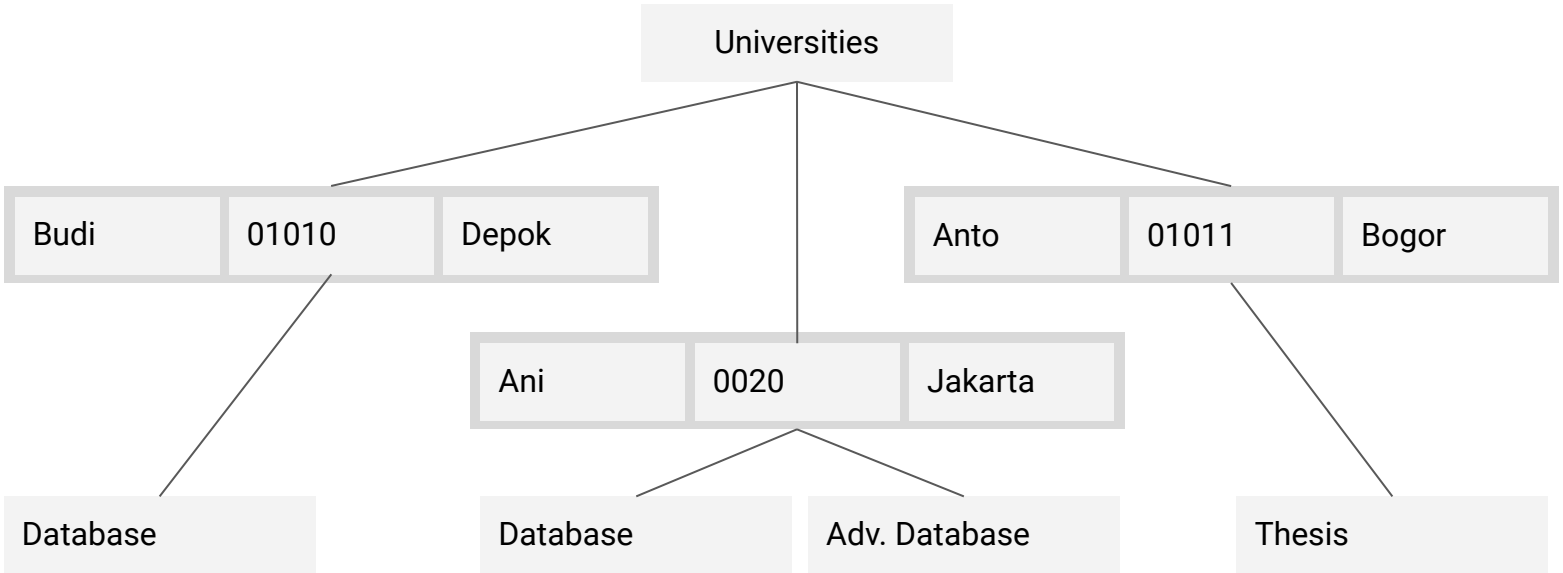


images: https://docs.oracle.com/cd/A87860_01/doc/appdev.817/a76976/adobjxmp.htm

Data Model Examples: Network Model



Data Model Examples: Hierarchical Model



Data Model Examples: Relational Model

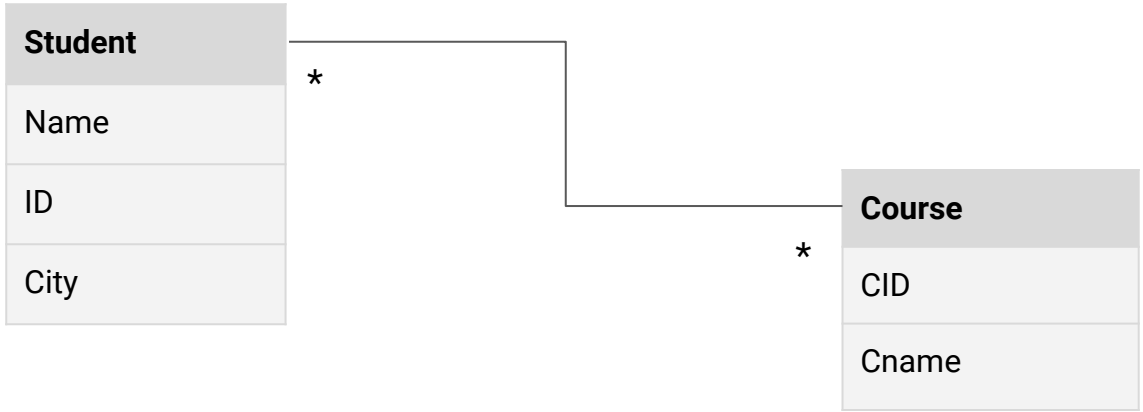
STUDENT

Name	ID	City	CID
Budi	01010	Depok	010
Ani	0020	Jakarta	010
Ani	0020	Jakarta	001
Anto	01011	Bogor	011

COURSE

CID	CName
010	Database
001	Adv. Database
011	Thesis

Data Model Examples: Object Oriented Model



Relational Model

Relational Model of Data Based on the Concept of a **Relation**

- ➔ Relation - a Mathematical Concept Based on **Sets**
- ➔ Strength of the Relational Approach to Data Management Comes From the Formal Foundation Provided by the Theory of Relations

RELATION: **A Table of Values**

- ➔ A Relation May Be Thought of as a **Set of Rows**
- ➔ A Relation May Alternately be Thought of as a Set of Columns
- ➔ Each Row of the Relation May Be Given an Identifier
- ➔ Each Column Typically is Called by its Column Name or Column Header or Attribute Name

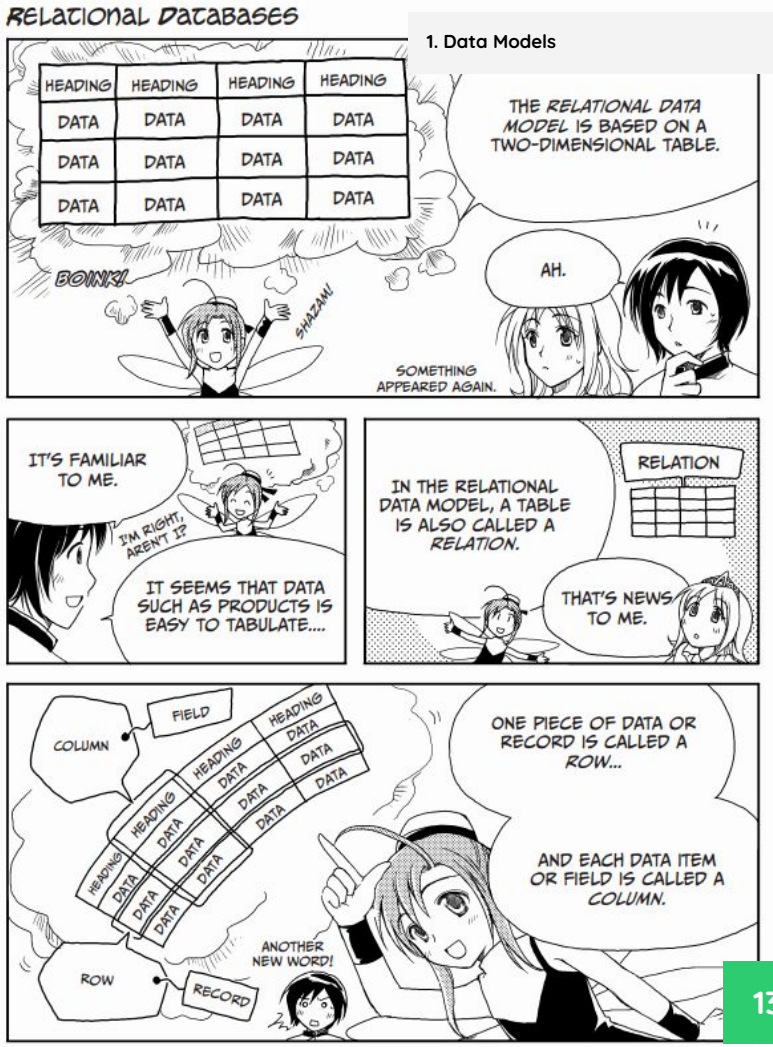


Illustration by: Takahashi & Azuma (2009)

Relational Tables - Rows, Columns, and Tuples

STUDENT	Name	StudentNumber	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

COURSE	CourseName	CourseNumber	CreditHours	Department
	Intro to Computer Science	CS1310	4	CS
	Data Structures	CS3320	4	CS
	Discrete Mathematics	MATH2410	3	MATH
	Database	CS3380	3	CS

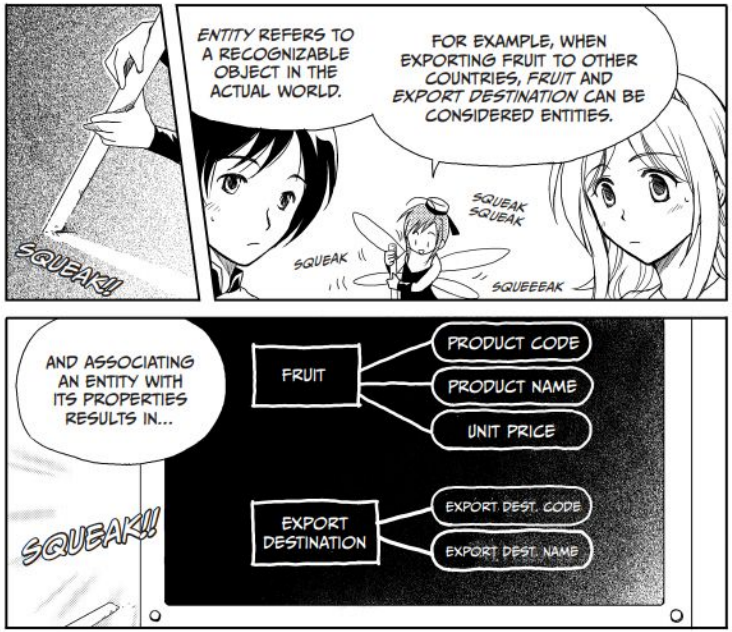
SECTION	SectionIdentifier	CourseNumber	Semester	Year	Instructor
	85	MATH2410	Fall	98	King
	92	CS1310	Fall	98	Anderson
	102	CS3320			
	112	MATH2410			
	119	CS1310			
	135	CS3380			

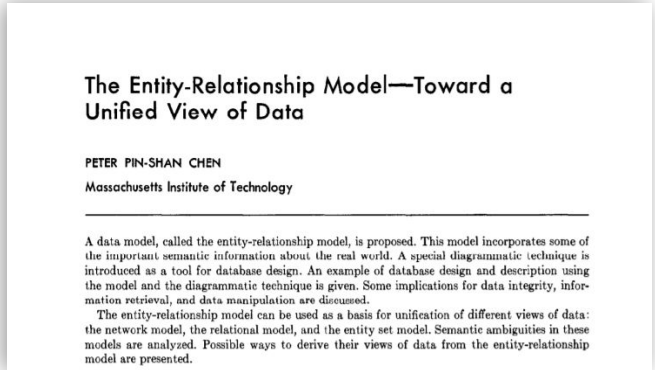
GRADE_REPORT	StudentNumber	SectionIdentifier	Grade
	17	112	B
	17	119	C
	8	85	A
	8	92	A
	8	102	B
	8	135	A

PREREQUISITE	CourseNumber	PrerequisiteNumber
	CS3380	CS3320
	CS3380	MATH2410
	CS3320	CS1310

Entity Relationship (ER) Data Model

- Originally Proposed by P. Chen, ACM TODS, Vol. 1, No. 1, March 1976
- Conceptual Modeling of Database Requirements
- Allows an Application's Information to be Characterized
- Basic Building Blocks are **Entities** and **Relationships**
- Well-Understood and Studied Technique
- Well-Suited for Relational Database Development
- Did Not Originally Include Inheritance





The Entity-Relationship Model—Toward a Unified View of Data

PETER PIN-SHAN CHEN

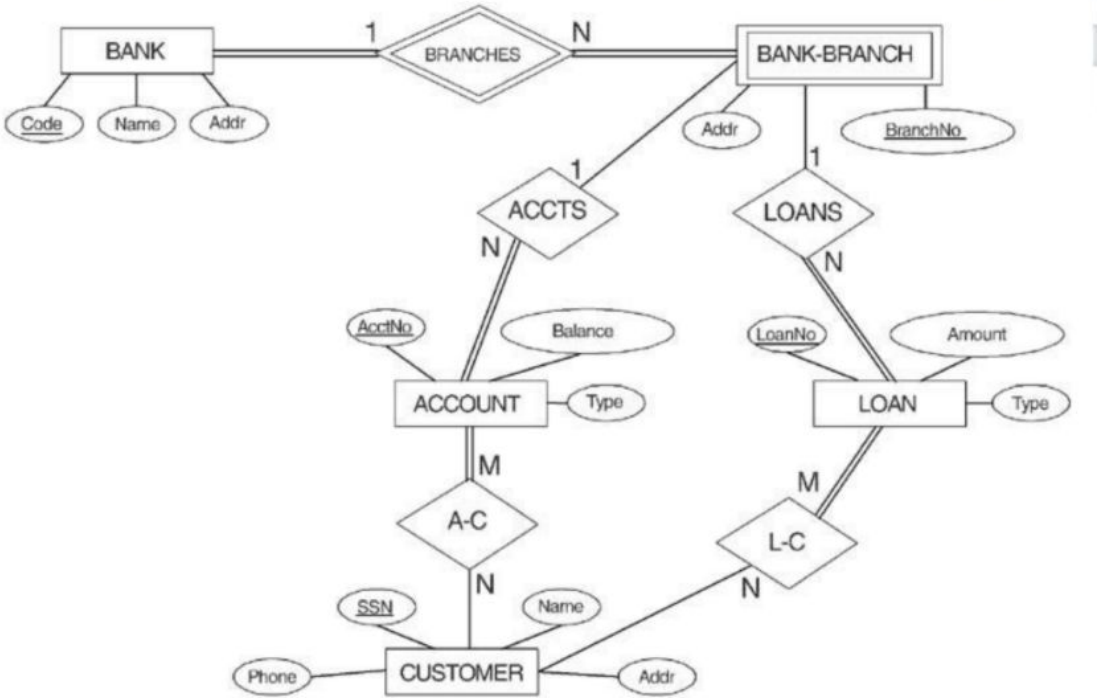
Massachusetts Institute of Technology

A data model, called the entity-relationship model, is proposed. This model incorporates some of the important semantic information about the real world. A special diagrammatic technique is introduced as a tool for database design. An example of database design and description using the model and the diagrammatic technique is given. Some implications for data integrity, information retrieval, and data manipulation are discussed.

The entity-relationship model can be used as a basis for unification of different views of data: the network model, the relational model, and the entity set model. Semantic ambiguities in these models are analyzed. Possible ways to derive their views of data from the entity-relationship model are presented.

Reference: Chen (1976); Takahashi & Azuma (2009)

ER Diagram


BANK
1
BRANCHES
N
BANK-BRANCH
Code
Name
Addr
Addr
BranchNo
1
ACCTS
1
LOANS
N
N
AcctNo
Balance
LoanNo
Amount
ACCOUNT
Type
LOAN
Type
M
M
A-C
L-C
N
N
SSN
Name
Phone
CUSTOMER
Addr

Outline

1. Data Models

- Categories of Data Models

- History of Data Models

2. Schema

- Three-Schema Architecture

3. DBMS Component

4. DBMS Architecture

Schema

Database Schema
The description of a database. Includes descriptions of the database structure and the constraints that should hold on the database.

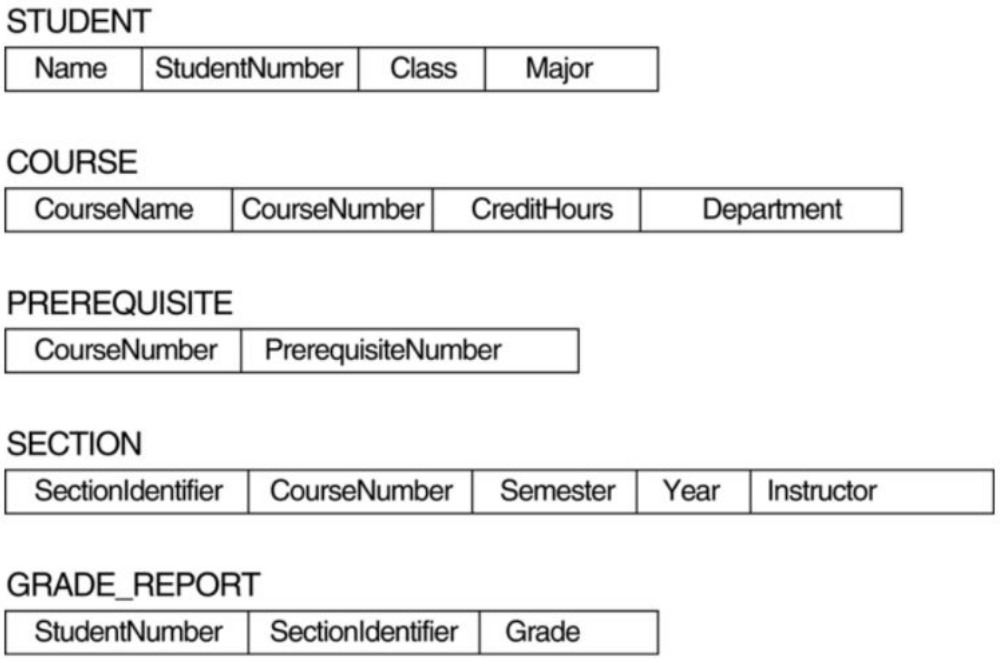
Schema Diagram
A diagrammatic display of (some aspects of) a database schema.

Schema Construct
A component of the schema or an object within the schema, e.g., STUDENT, COURSE.

Database State/Snapshot
The actual data stored in a database at a particular moment in time. Also called the current set of occurrences/instances).

images: https://medium.com/@tiwarivarchasva1251/guidelines-for-creating-a-good-database-schema-part-1-54f6849cea61

Schema Diagram



Database Schema Vs. Database State

Database Schema

- The database schema **changes very infrequently**
- Schema is also called **intension**

VS

Database State

- The database state **changes every time** the database is updated.
- State is called **extension**.

Database state: Refers to the **content** of a database at a moment in time.

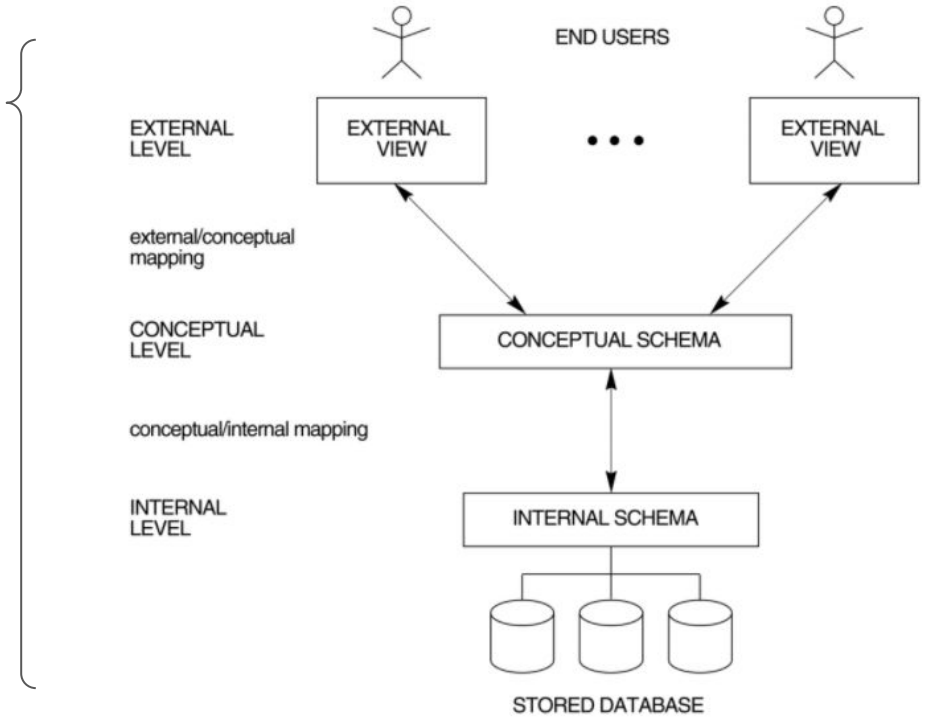
Initial Database State: Refers to the database when it is loaded

Valid State: A state that satisfies the structure and constraints of the database.

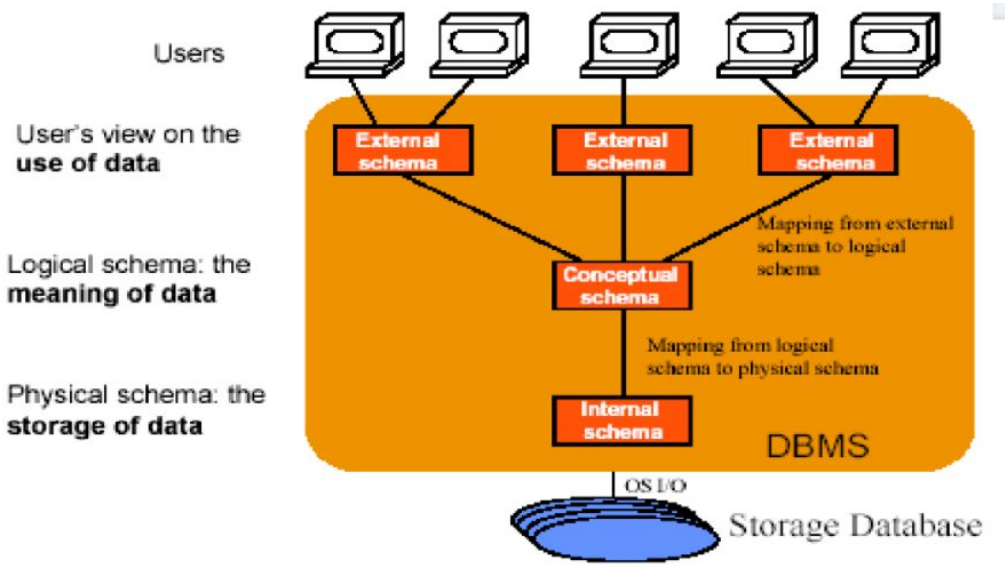
Three-Schema Architecture

Proposed to support DBMS characteristics of:

- ➔ **Program-data independence**
- ➔ Support of **multiple views** of the data



Three-Schema Architecture: Another View



Three-Schema Architecture

Defines DBMS schemas at **three levels**:

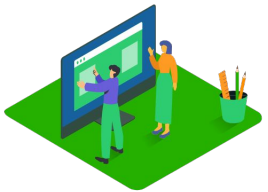
1 **Internal schema**
at the internal level to describe physical storage structures and access paths. Typically uses a physical data model.

2 **Conceptual schema**
at the conceptual level to describe the structure and constraints for the whole database for a community of users. Uses a conceptual or an implementation data model.

CUSTOMER

Name	Addr	Sex	Age

3 **External schema**
at the external level to describe the various user views. Usually uses the same data model as the conceptual level.



Conceptual Schema

Describes the **meaning of data** in the universe of discourse

- ➔ Emphasizes on general, conceptually relevant, and often time invariant **structural aspects** of the universe of discourse
- ➔ Excludes the physical organization and access aspects of the data

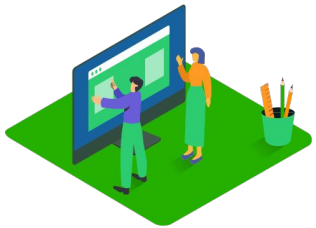
CUSTOMER

Name	Addr	Sex	Age

External Schema

Describes parts of the information in the conceptual schema in a form convenient **to a particular user group's view**

➔ Derived from the conceptual schema



CUSTOMER

Name	Addr	Sex	Age

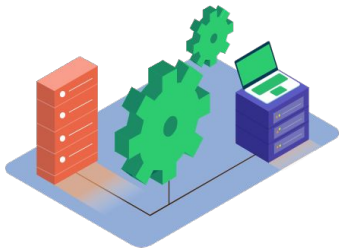
MALE_CUSTOMER (Name, Addr) =
CUSTOMER (Name, Addr, Sex, Age)
WHERE Sex = M

MALE_CUSTOMER

Name	Addr

Internal Schema

Describes how the information described in the conceptual schema is **physically represented** in a database to provide the overall best performance



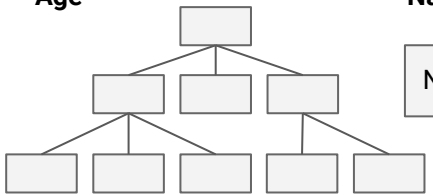
CUSTOMER

Name	Addr	Sex	Age

CUSTOMER

Name	Addr	Sex	Age

B^+-tree on Age



index on Name

Name	Ptr

Block size = 100

Unified Example of Three Schema

An Example Query:

"List all employees whose has more than 5 years working experience?"

```
SELECT  e.ENAME, e.DEPT, e.EXP
FROM  EMP e
WHERE  e.EXP > 5 year.
```

External Schema:

```
CREATE EMP(ENAME, DEPT, EXP)
        AS VIEW OF EMPLOYEE(EN, DNO, EXP_YEAR)
CREATE PAYROLL(EN, SAL, SSN, BirthDate)
        AS VIEW OF EMPLOYEE(SSN,EN,SALARY,BDATE)
```

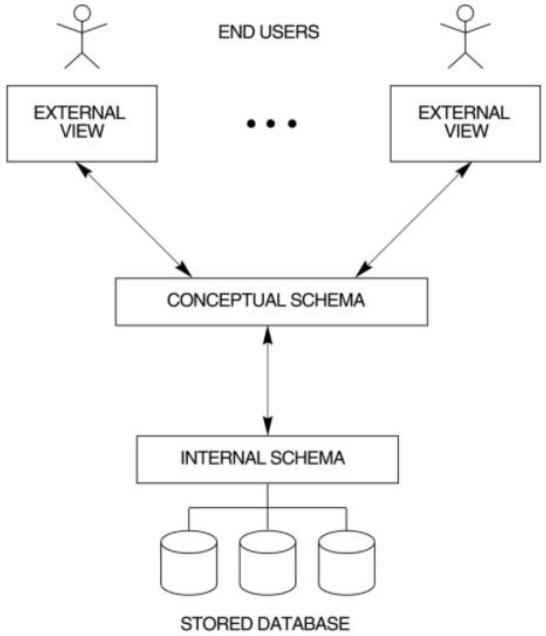
Conceptual Schema:

EMPLOYEE(SSN, EN, DNO, SALARY, EXP_YEAR, BDATE, STARTDATE)

Internal Schema:

Cluster Index on SNN;

No-cluster B-tree Indexes on DNO, EXP_YEAR, STARTDATE.



Data Independence

> Ability that allows application programs **not being affected by changes** in irrelevant parts of the conceptual data representation, data storage structure and data access methods

- ➔ **Invisibility** (transparency) of the details of entire database organization, storage structure and access strategy to the users
- ➔ Both logical and physical

Recall software engineering concepts:

- ➔ **Abstraction** the details of an application's components can be hidden, providing a broad perspective on the design
- ➔ **Representation independence**: changes can be made to the implementation that have no impact on the interface and its users

Data Independence (Continued)

Logical Data Independence

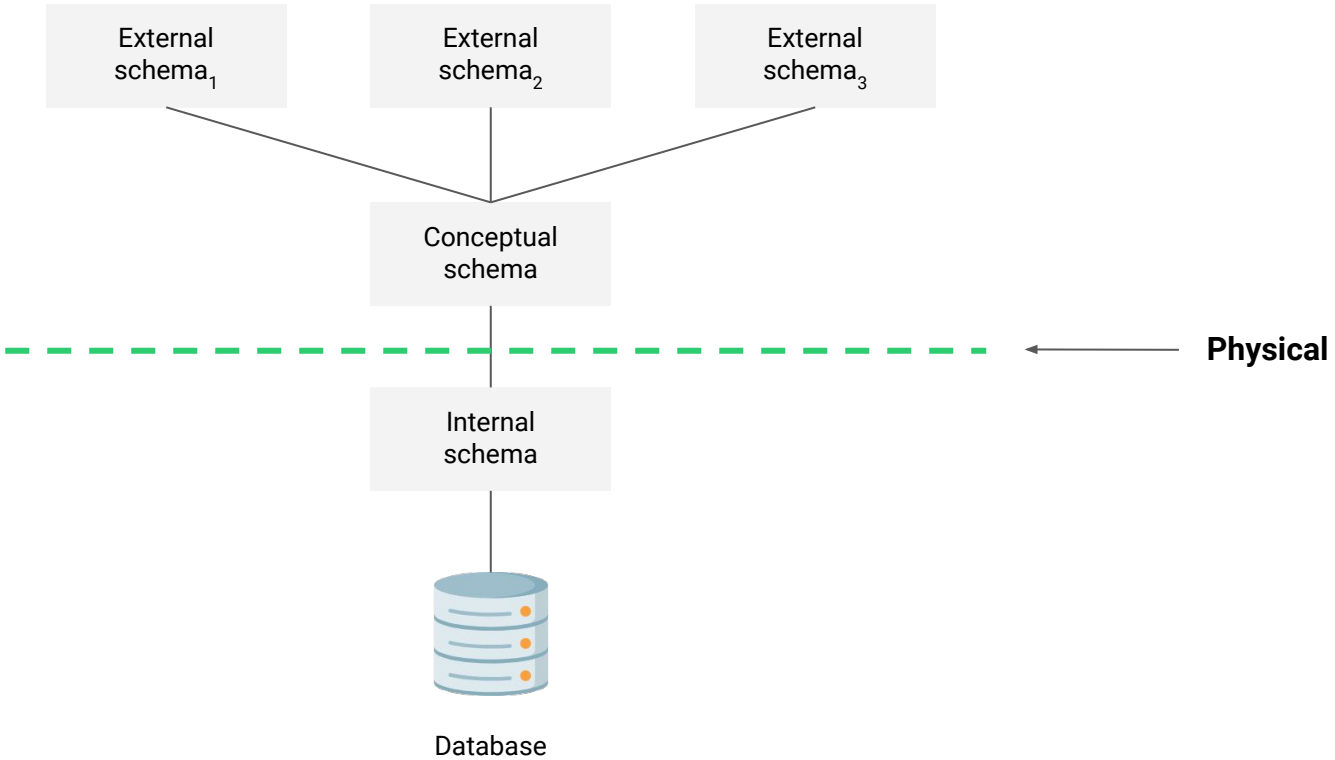
The capacity to change the conceptual schema without having to change the external schemas and their application programs.

Physical Data Independence

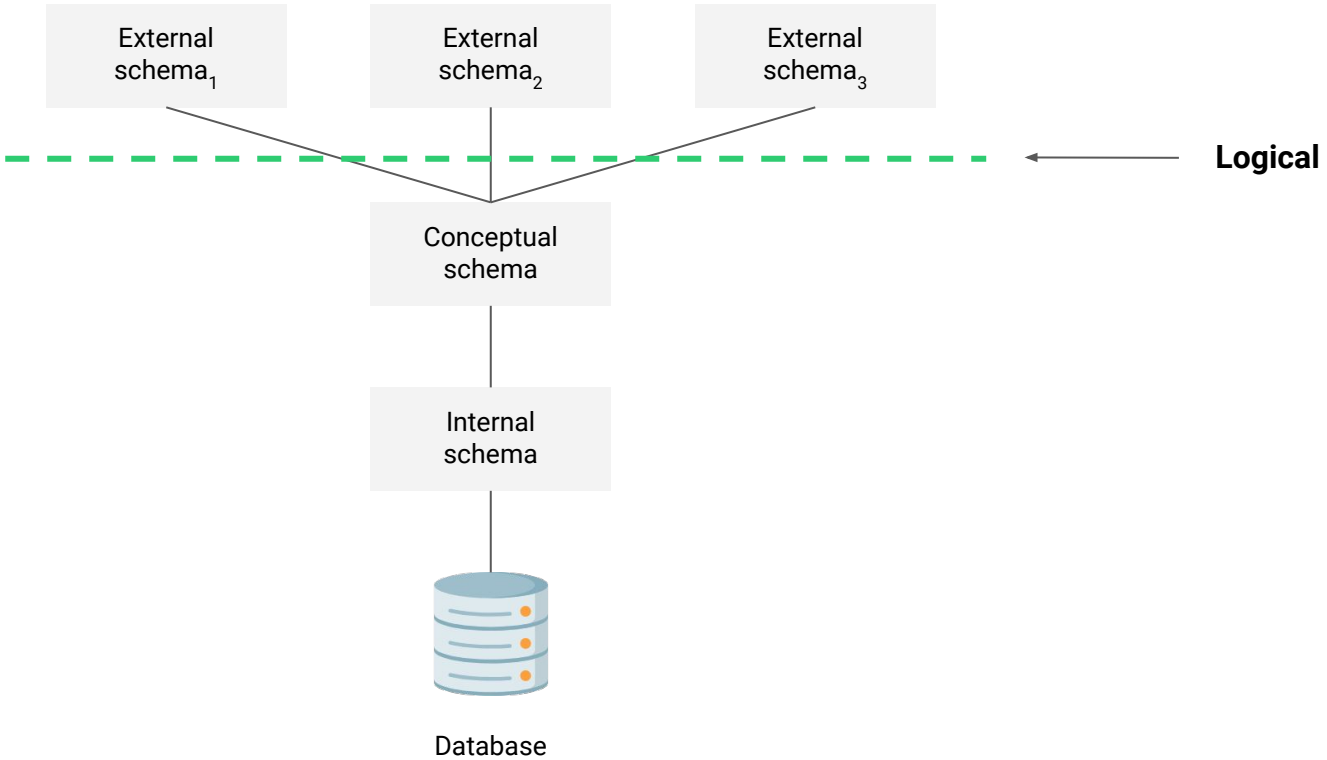
The capacity to change the internal schema without having to change the conceptual schema.

When a schema at a lower level is changed, only the mappings **between this schema and higher-level** schemas need to be changed in a DBMS that fully supports data independence. The higher-level schemas themselves are unchanged. Hence, the application programs need not be changed since they refer to the external schemas.

Physical Data Independence



Logical Data Independence



Outline

1. Data Models

- Categories of Data Models

- History of Data Models

2. Schema

- Three-Schema Architecture

3. DBMS Component

4. DBMS Architecture

DBMS Languages

Data Definition Language (DDL)

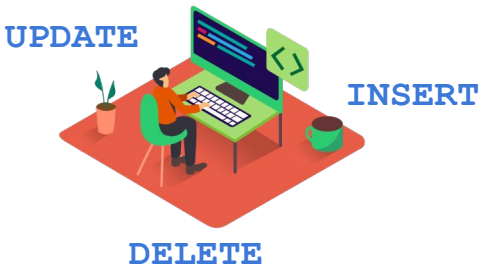
Used by the DBA and database designers to **specify** the conceptual schema and internal schema of a database and any mapping between the two.

- ➔ In many DBMSs where a clear separation of conceptual and internal schema, DDL is used to define conceptual schema only. **Storage definition language (SDL)** define the internal schema and **view definition language (VDL)** are used to define user view and their mapping to the conceptual schemas.
- ➔ Most DBMSs, the DDL is used to define both conceptual and external schemas

DBMS Languages (Continued)

Data Manipulation Language (DML)

Used to specify database **retrievals** and **updates**.



- ➔ IDML commands (data sublanguage) can be embedded in a general-purpose programming language (host language), such as COBOL, C or an Assembly Language.
- ➔ Alternatively, stand-alone DML commands can be applied directly (**query** language).

DBMS Languages (Continued)

1

High Level or Non-procedural Languages:

e.g., SQL, are set-oriented and specify what data to retrieve than how to retrieve. Also called declarative languages.

2

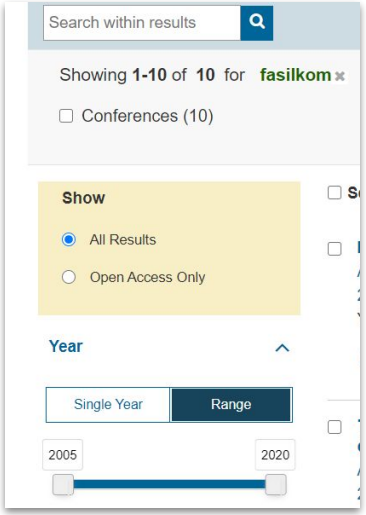
Low Level or Procedural Languages:

record-at-a-time; they specify how to retrieve data and must be embedded in programming language

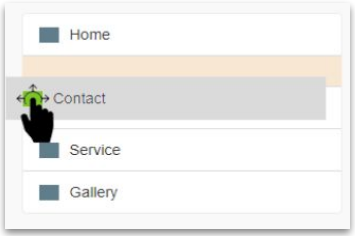
DBMS Interfaces

Menu-based
Popular for browsing on the web

Forms-based
Designed for naïve users



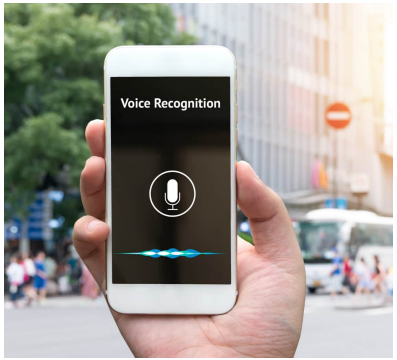
Natural language
requests in written English: → “Show the student that have GPA above 3.0”



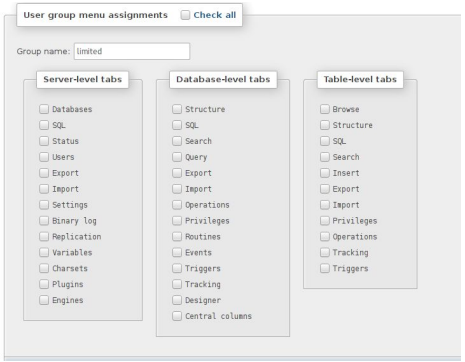
Graphics-based
Point and Click, Drag and Drop etc.

Combinations of various types

Other DBMS Interfaces



Speech as Input/Output



Interfaces for the DBA

- ➔ Creating accounts, granting authorizations
- ➔ Setting system parameters
- ➔ Changing schemas or access path

Parametric interfaces

(e.g., bank tellers) using function keys.

The Database System Environment

Main DBMS modules:

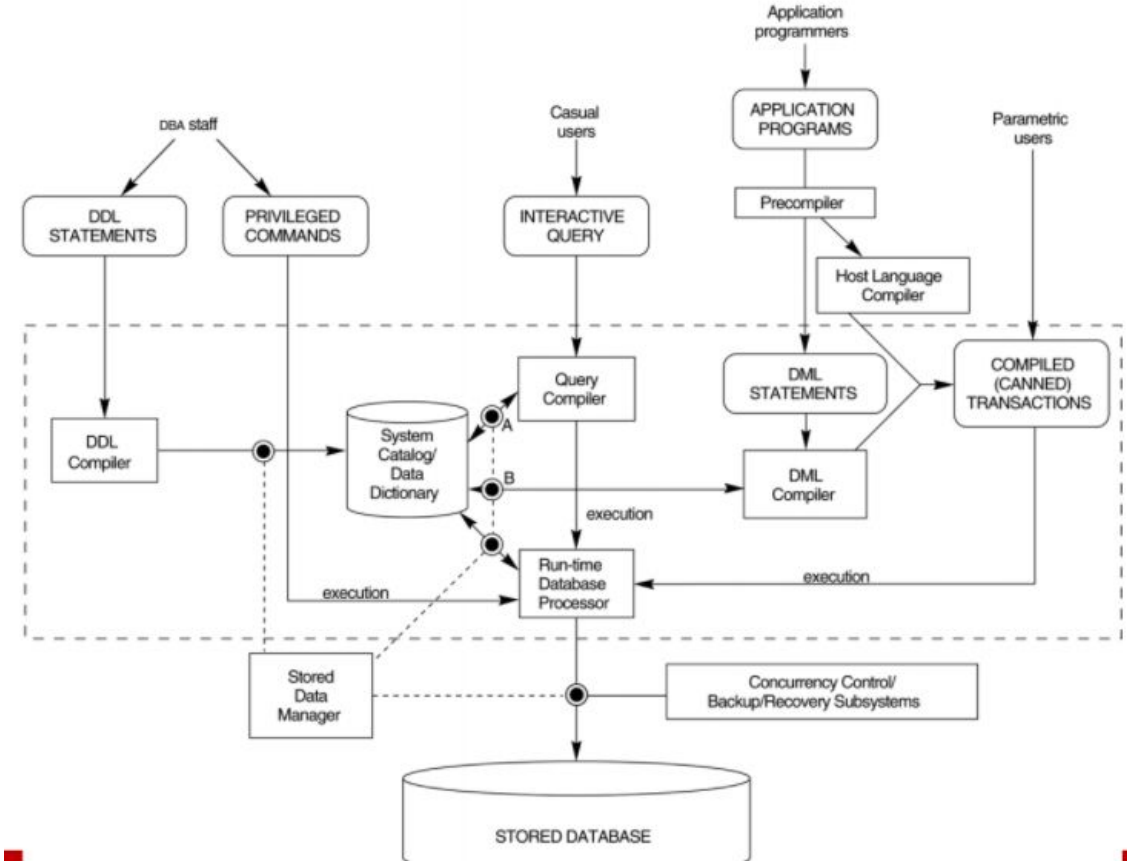
- DDL Compiler
- DML Compiler
- Ad-hoc (interactive) query compiler
- Run-time database processor
- Stored data manager
- Concurrency/back-up/recovery subsystem

DBMS utility modules:

- Loading routines
- Backup utility

...

Component modules of a DBMS and their interactions



Database System Utilities

To perform certain functions such as:

- ➔ **Loading** data stored in files into a database. Includes data conversion tools.
- ➔ **Backing up** the database periodically on tape.
- ➔ **Reorganizing** database file structures.
- ➔ **Report generation** utilities.
- ➔ **Performance monitoring** utilities.
- ➔ Other functions, such as **sorting**, **user monitoring**, **data compression**, etc.

Other Tools

Data dictionary/repository

Used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc.

Application Development Environments and CASE (computer-aided software engineering) tools

- ➔ Power builder, Builder, VB, Java, C, C++, etc.
- ➔ Ms. Visio, ER-Win, DBDesigner, etc.

Outline

1. Data Models

- Categories of Data Models

- History of Data Models

2. Schema

- Three-Schema Architecture

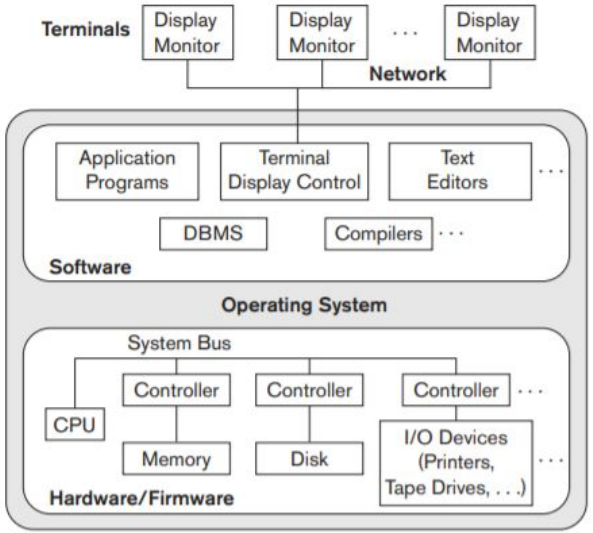
3. DBMS Component

4. DBMS Architecture

Centralized Architecture

Centralized DBMS

combines everything into single system (PC) including- DBMS software, hardware, application programs and user interface processing software.



Client-Server Architectures

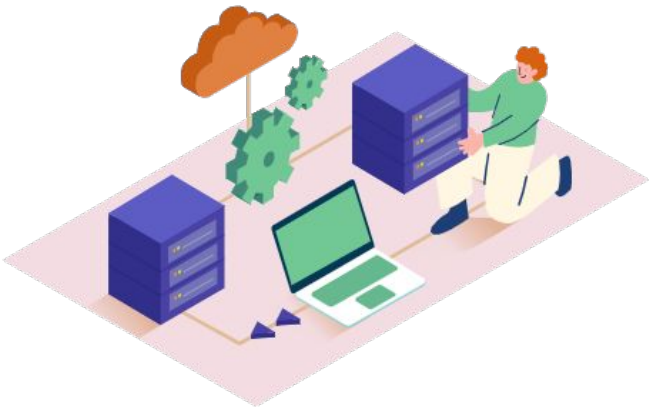
Servers

Specialized Servers with Specialized functions.
Ex. Database Server, File Server, Web Server, Email Server

Client

Provide appropriate interfaces and a client-version of the system to access and utilize the server resources.

- ➔ Clients maybe **diskless machines** or **PCs** or **Workstations** with disks with only the client software installed.
- ➔ **Connected** to the servers via some form of a network (LAN: local area network, wireless network, etc.)

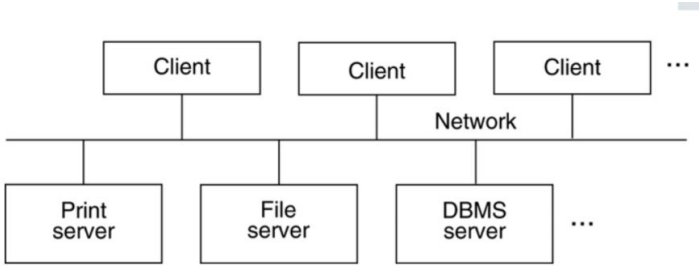


Two Tier Client-Server Architecture

User Interface Programs and Application Programs

run on the **client** side

➔ Interface called **ODBC** (Open Database Connectivity) provides an Application program interface (API) allow client side programs to call the DBMS. Most DBMS vendors provide ODBC drivers.



Logical two-tier client/server architecture

Three Tier Client-Server Architecture

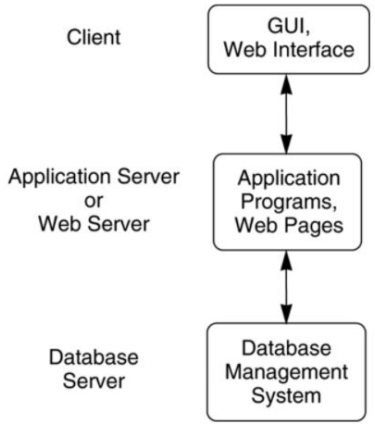
Common for **Web applications**

Intermediate Layer called **Application Server** or **Web Server**:

- ➔ Stores the web connectivity software and the rules and business logic (constraints) part of the application used to access the right amount of data from the database server
- ➔ Acts like a conduit for sending partially processed data between the database server and the client.

Additional Features-**Security**:

- ➔ Encrypt the data at the server before transmission
- ➔ Decrypt data at the client



Logical three-tier client/server architecture

Database Classification

According to the data models	• object-oriented DBMS (ObjectStore, Ontos, etc.) • **relational DBMS** (Oracle, Sybase, Informix, DB2, Microsoft SQL server etc.) • network DBMS (DBTG ...) • hierarchical DBMS (IMS ...)
According to the number of users	• single-user DBMS (mainly for PCs) • multi-user DBMS
According to the number of sites	• centralized DBMS (Oracle, Sybase, etc.) • distributed DBMS (R*, ...) • federated DBMS homogeneous DBS heterogeneous DBS
According to the types of access methods	• general purpose DBMS • special purpose DBMS

References

Images:
https://engineering.linecorp.com/en/blog/bot-designer-for-previewing-conversation-with-chatbots/
https://laravelarticle.com/laravel-6-drag-and-drop-menu-sorting
https://www.athreon.com/benefits-and-challenges-of-speech-recognition/
https://docs.phpmyadmin.net/en/latest/privileges.html
https://www.sciencephoto.com/media/111313/view/cdc-6400-mainframe-computer
https://www.costco.co.uk/Business-Office-Supplies/Office-Electronics/Cash-Registers/Casio-Cash-Register-Black-SR-C4500MD-BK/p/304712

Q&A