Three-Schema Architecture & Data Independence

Database Design

Department of Computer Engineering

Sharif University of Technology

Maryam Ramezani maryam.ramezani@sharif.edu

# Three-Schema Architecture

Important Characteristics of the Database

❑ (1) Use of a catalog to store the database description (schema) so as to make it self-describing

❑ (2) Insulation of programs and data (program-data and program-operation independence),

❑ (3) Support of multiple user views.

We need an architecture to separate the user applications from the physical database.

Three-schema Architecture
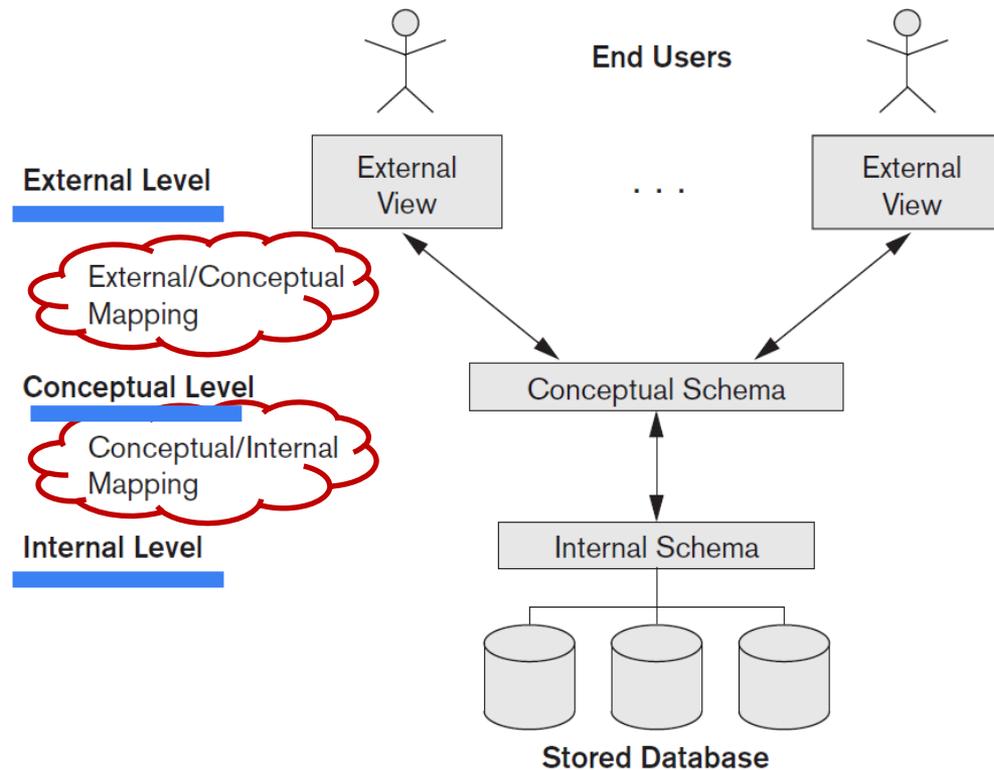
- ❑ Mapping is used to transform the request and response between various database levels of architecture.
- ❑ Mapping is not good for small DBMS because it takes more time.
- ❑ In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema.
- ❑ In Conceptual / Internal mapping, DBMS transform the request from the conceptual to internal level.

❑ The main objective of three level architecture is to enable multiple users to access the same data with a personalized view while storing the underlying data only once.

❑ It separates the user's view from the physical structure of the database for the following reasons:

- o Different users need different views of the same data.
- o The users of the database should not worry about the physical implementation and internal workings of the database such as data compression and encryption techniques, hashing, optimization of the internal structures etc.
- o All users should be able to access the same data according to their requirements.
- o DBA should be able to change the conceptual structure of the database without affecting the users.
- o Internal structure of the database should be unaffected by changes to physical aspects of the storage.

1. Internal Level

❑ The internal level has an internal schema which describes the physical storage structure of the database.

❑ The internal schema is also known as a physical schema.

❑ It uses the physical data model. It is used to define that how the data will be stored in a block.

❑ The physical level is used to describe complex low-level data structures in detail.

Internal view

STORED_EMPLOYEE record length 60

Empno : 4 decimal offset 0 unique
Ename : String length 15 offset 4
Salary : 8,2 decimal offset 19
Deptno : 4 decimal offset 27
Post   : string length 15 offset 31

1. Internal Level

❑ The internal level is generally is concerned with the following activities:

- ○ Storage space allocations.

  For Example: B-Trees, Hashing etc.

- ○ Access paths.

  For Example: Specification of primary and secondary keys, indexes, pointers and sequencing.

- ○ Data compression and encryption techniques.

- ○ Optimization of internal structures.

- ○ Representation of stored fields.

2. Conceptual Level

❑ The conceptual schema describes the design of a database at the conceptual level.

❑ Conceptual level is also known as logical level.

❑ The conceptual schema describes the structure of the whole database.

❑ The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.

❑ In the conceptual level, internal details such as an implementation of the data structure are hidden.

❑ Programmers and database administrators work at this level.

Global view

```
EMPLOYEE

Empno  :  Integer(4) Key
Ename  :  String(15)
Salary  :  String (8)
Deptno :  Integer(4)
Post   :  String (15)
```

3. External Level

❑ At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.

❑ An external schema is also known as view schema.

❑ Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.

❑ The view schema describes the end user interaction with database systems.

External View

Empno	Ename

Empno	Ename	Salary	DeptNo

Mapping Between Levels

The three levels of DBMS architecture don't exist independently of each other. There must be correspondence between the three levels.

❑ **External/ Conceptual Mapping**
  ○ lies between the external level and the Conceptual level. Its role is to define the correspondence between a particular external and the conceptual view.

❑ **Conceptual/ Internal Mapping**
  ○ lies between the conceptual level and the internal level. Its role is to define the correspondence between the records and fields of the conceptual level and files and data structures of the internal level.

Data Independence

Mapping Between Levels

❑ Data independence can be explained using the three-schema architecture.

❑ Data independence refers characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level.

❑ If new data structures are added to the database or existing structures in the database are modified then the application programs that use the database are unaffected, provided they don't directly depend upon what has been modified.

  ○ For example, if we add a new column to a record or create a new table, existing applications are unaffected. However, if we remove a column from a table that an application program uses, then that application program is affected by this change and must be modified accordingly. We don't consider this remove case in Data Independence.

Data Independence

❑ There are two types of data independence:
  ○ Physical Data Independence (PDI)
  ○ Logical Data Independence (LDI)

Physical Data Independence (PDI)
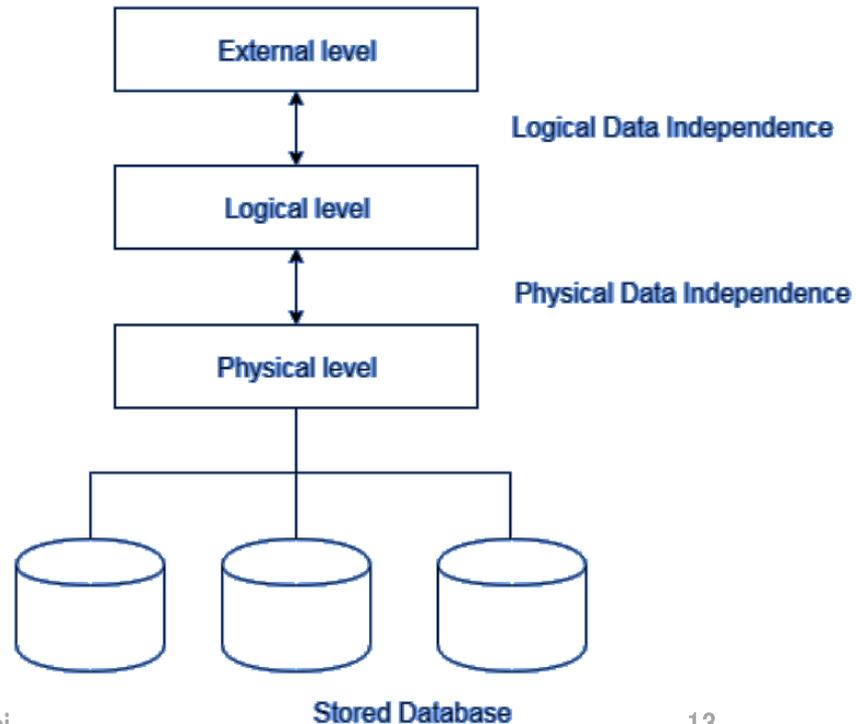
❑ Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.

❑ If we do any changes in the storage size of the database system server, then the conceptual structure of the database will not be affected.

❑ Physical data independence is used to separate conceptual levels from the internal levels.

❑ Physical data independence exists in most databases and file environments where physical details, such as the exact location of data on disk, and hardware details of storage encoding, placement, compression, splitting, merging of records, using different file organizations or modifying indexes and so on are hidden from the user. Applications remain unaware of these details.

Logical Data Independence (LDI)

❑ Logical data independence refers characteristic of being able to change the conceptual schema without having to change the external schema.

❑ Logical data independence is used to separate the external level from the conceptual view.

❑ If we do any changes in the conceptual view of the data, then the user view of the data would not be affected.

❑ Logical data independence occurs at the user interface level.

❑ Changes to constraints can be applied to the conceptual schema without affecting the external schemas or application programs.

❑ Logical data independence is harder than PDI to achieve because it allows structural and constraint changes without affecting application programs—a much stricter requirement.

Logical Data Independence (LDI)

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

TRANSCRIPT

Student_name	Student_transcript				
	Course_number	Grade	Semester	Year	Section_id
Smith	CS1310	C	Fall	08	119
	MATH2410	B	Fall	08	112
Brown	MATH2410	A	Fall	07	85
	CS1310	A	Fall	07	92
	CS3320	B	Spring	08	102
	CS3380	A	Fall	08	135

GRADE_REPORT

Student_number	Student_name	Section_identifier	Course_number	Grade
17	Smith	112	MATH2410	B
17	Smith	119	CS1310	C
8	Brown	85	MATH2410	A
8	Brown	92	CS1310	A
8	Brown	102	CS3320	B
8	Brown	135	CS3380	A

Logical Data Independence (LDI)

❑ Redesign the database: When a table is converted to two different tables:

    ○ Solution: View Definition on view
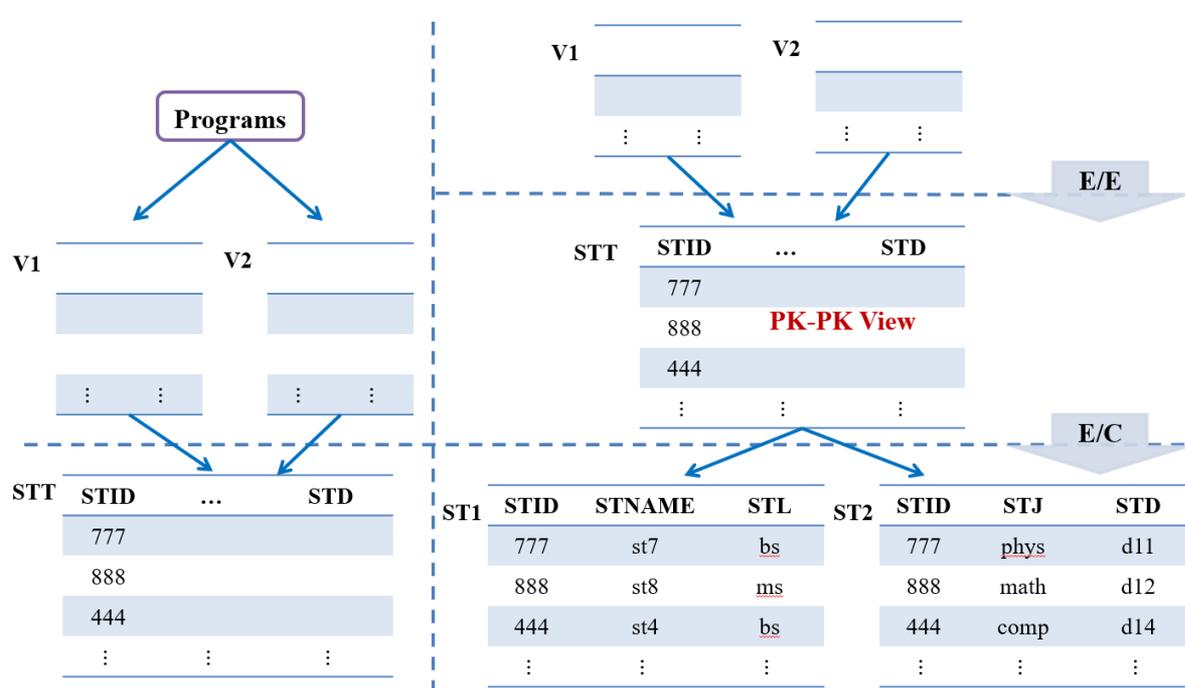
CREATE TABLE ST1
        (STID ...,
        ...
        STL ...**)**
        PRIMARY KEY STID

CREATE TABLE ST2
        (STID ...,
        ...
        STD ...**)**
        PRIMARY KEY STID

INSERT INTO ST1
        (SELECT STID, STNAME, STLEV
        FROM STT**)**

INSERT INTO ST2
        (SELECT STID, ..., STD
        FROM STT**)**

(Data Migration)

CREATE VIEW STT
        AS SELECT STID, ..., STDEID
        FROM ST1 **JOIN** ST2

DROP TABLE STT

Problems:
o In data catalog STT is a table not a view!
o Creating view with the same name of existing table is rejected!!! Renaming is needed!
o Updating on view STT is not accepted by DBMS.
              Solution? Trigger!!

Summary

❑ Whenever we have a multiple-level DBMS, its catalog must be expanded to include information on how to map requests and data among the various levels. The DBMS uses additional software to accomplish these mappings by referring to the mapping information in the catalog.

❑ Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed. Hence, application programs referring to the higher-level schema need not be changed.

❑ The physical schema is hidden beneath the logical schema and can usually be changed easily without affecting application programs. Application programs are said to exhibit physical data independence if they do not depend on the physical schema and thus need not be rewritten if the physical schema changes.

References

❑ Chapter 2: Part2: FUNDAMENTALS OF Database Systems