VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

Introduction to Database Systems
Part II

Coordinator: Dr Hui Ma
Lecturer: Dr Pavle Mogin

SWEN 304
Database System Engineering

Plan for Intro to DB Systems II

- Data models
  - Structural component
  - Dynamic component
- Data manipulation languages
  - Navigational
  - Declarative
- Schemas and Instances
- The Three Schema Architecture
- Program - Data Independence

  - *Reading:*
    - *Chapter 2 of the textbook*

Data Models

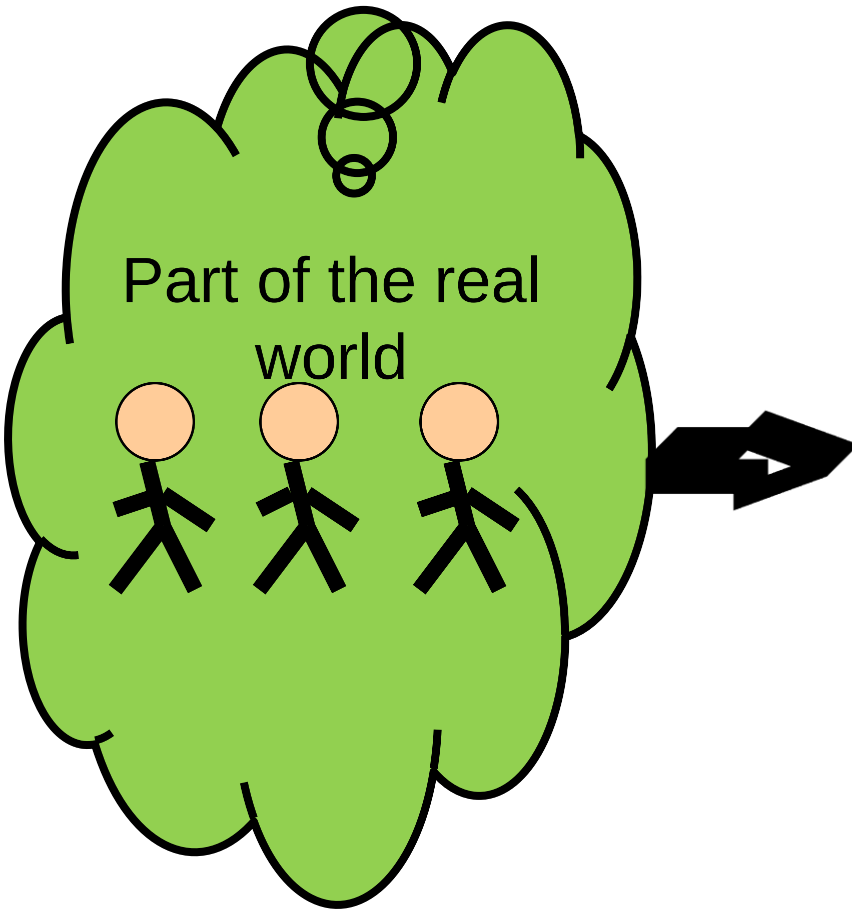
- A Data Model: a set of concepts to describe the structure and constraints of a database, and the operations for manipulating these structures
- Also, a data model is said to be a mathematical abstraction that is used:
  - To make an approximate representation (an abstraction) of a real system, and
  - To make a model of a database of the real system
- A data model provides components to represent:
  - The structure (and constraints), and
  - Dynamics

  of a real system and to map them to a database structure, constraints, and operations

Data Models – Structural Component

- The structural component contains primitive concepts, and rules to combine them into more complex concepts
- Example:
  - Use data that describe some properties of an entity to build the model of the entity itself
  - (Surname: Bond, Name: James, StudentId: 007007, Major: Comp)
- (Integrity) constraints are statements about values and relationships that must hold between data
- Examples:
  - Each student must have unique StudId,
  - NoofPts is a nonnegative integer, less than 1000,
  - For any course, a student can get at most one grade in a term

Structural Component (continued)



Abstract representation at the level of properties

Student

StId	*Name*	*Surname*	*Pts*

Constraint:
StId values are unique

Dynamic Component - Operations

- Dynamic aspects of a real world system (UoD) are modeled using:
  - Basic operations (database retrieval and update), and
  - Complex operations (complex logic, GUI)
- These operations are supported by programming languages:
  - A data manipulation language (retrieval and updates), and
  - A general purpose programming language (complex logic and GUI)

Data Manipulation Language (DML)

- Used to retrieve, insert, delete, and modify data
  - In the real world, entities are created (insert), used (retrieve) and altered (update), and finally they expire (delete)
- Always selects a relatively small part of a database and transfers it from disk to main memory
- A DML can be:
  - Either navigational, or
  - Declarative

Navigational DML

- Procedural (has loops, branching conditions),
- Selects a record at a time,
- Programmer explicitly utilizes information about database physical organization to "navigate" through the database,
- Programmer defines WHAT and HOW

Declarative DML

- Non procedural,
- Set oriented (selects all data that match given conditions),
- Search conditions are defined according to an abstract database representation, so the programs are completely independent of a database physical organization,
- A programmer (or even a casual user) has to define just WHAT

Navigational versus Declarative DML (1)

- Query: “Retrieve all Courses and Grades of the student with StudentId = 131313”
- Simplified navigational pseudo code:

```
Find the student record with StudentId = 131313 in GRADES
If successful then
    Do while there are courses connected to the student
        Find next course Id in GRADES
        Find corresponding Grade
        Find corresponding Course name in COURSE
    End do
Else
    Display error message
End if
```

Navigational versus Declarative DML (2)

- Query: “Retrieve all Courses and grades of the student with StudentId = 131313”

- A fully declarative program:

```
SELECT Course_id, Cname, Grade
FROM COURSE C, GRADES G
WHERE StudentId = 131313 AND G.CourseId
= C.CourseId;
```

Database Users and Languages

- Database administrator (DBA) and DB designer:
  - Data Definition Language (DDL) to describe conceptual (or implementation) schema,
  - View Definition Language (VDL) to describe user views
  - Storage Definition Language (SDL) to describe internal schema
- Casual end users:
  - Interactive declarative DML – mainly for database queries
- Naïve (or parametric) users:
  - Canned transaction programs (written by programmers)
- Programmers:
  - Interactive DML,
  - DML embedded into general purpose (host) programming language

Classification of Data Models

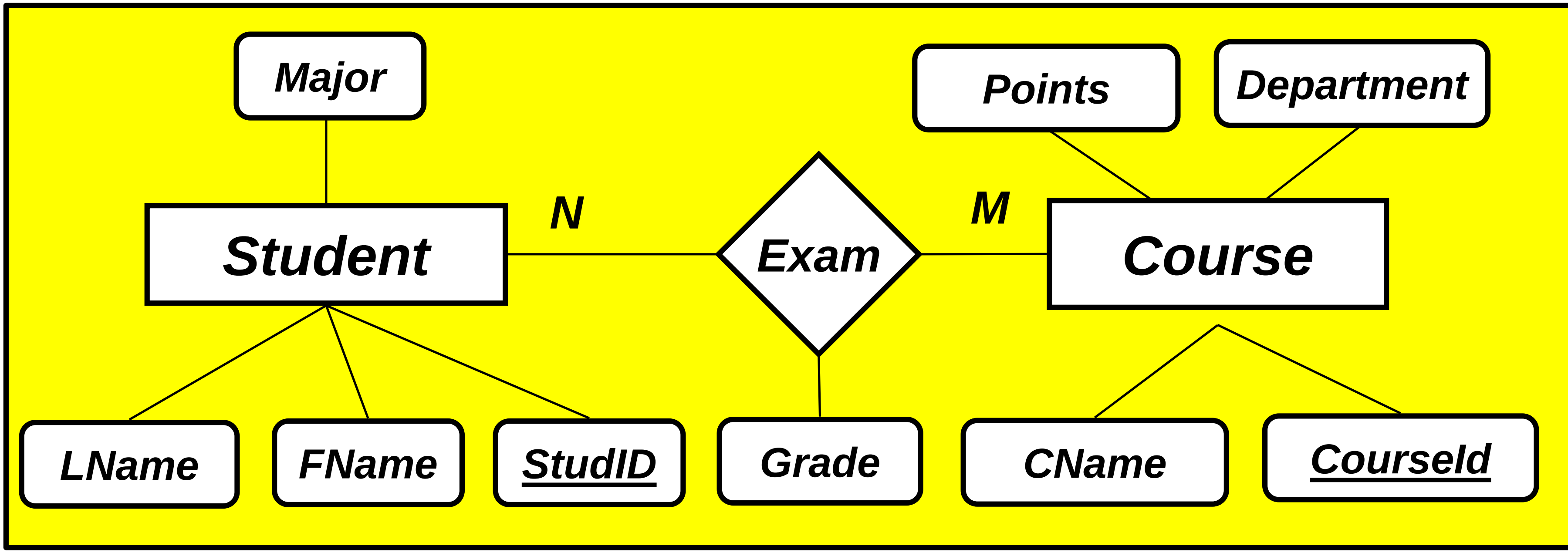
- High level (conceptual) data models:
  - Enhanced Entity Relationship (EER - highly abstract, no DBMS),
- Higher level implementation data models:
  - Object-oriented (some DBMS, mainly navigational), used as a conceptual data model in the software engineering
- Representational (Implementation) data models
  - network (legacy, navigational),
  - hierarchical (legacy, navigational),
  - relational (contemporary DBMS mainly implemented on it, declarative),
- Low level (physical storage) data models
  - ISAM, VSAM (file systems)

Schemas and Instances

- A schema is a model of a real system, and an abstract description of it’s database
- A database schema bears information about database structure and constraints
- A database schema is a fairly static category
- An object of a database schema (as *Student*) is sometimes called schema construct
- A database schema is designed and then defined to a DBMS using a Data Definition Language

An ER and a Relational Database Schema

ER Schema



Relational Schema

STUDENT (StudentId, LName, FName, Major),
GRADES (StudentId, CourseId, Grade),
COURSE (CName, CourseId, Points, Department)

Schemas and Instances (continued)

- A database state or a schema instance (often referred to as a database) is a collection of related data that mirror one of the possible states of a real system
- A schema instance:
  - Is produced by populating (loading) schema description by data
  - Corresponds to schema structure, and
  - Satisfies all schema constraints
- A schema instance should reflect changes of the real system's state, and that is accomplished by its updating (a schema instance is a dynamic category)
- Often, schemas are referred to as ***intensions*** and instances are referred to as ***extensions***

A Database Instance

Student

LName	*FName*	*StudId*
Smith	Susan	131313
Bond	James	007007
Smith	Susan	555555

Course

CName	*CourseId*
DB Sys	C302
SofEng	C301
DisMat	M214

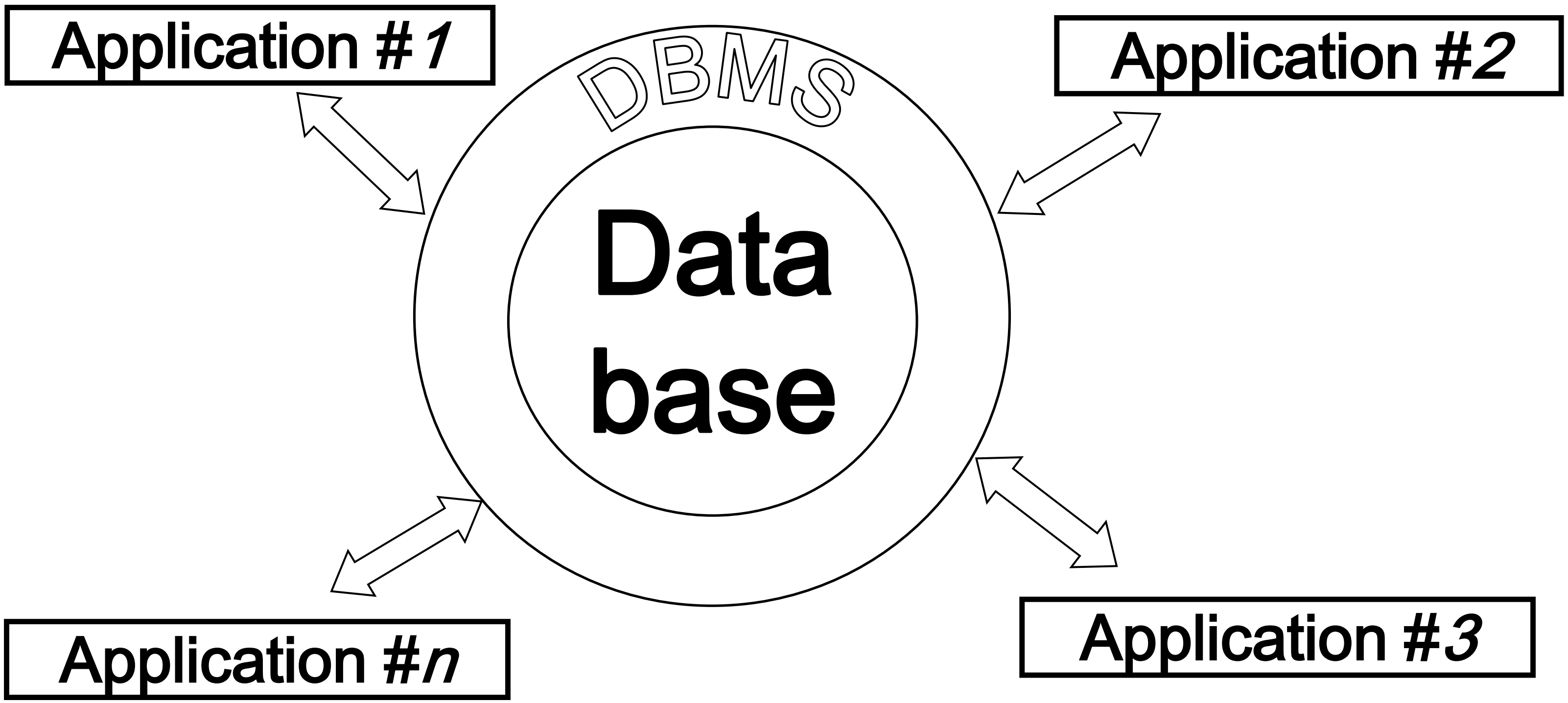
Grade

StudId	*CourseId*	*Grade*
007007	C302	A+
007007	C301	A
007007	M214	A+
131313	C301	B-
555555	C301	C
131313	C302	D
555555	C302	E

Data Organization Approaches

- There are two characteristic approaches to data organization and storage in the realm of an information system
- These are:
  - File approach (traditional), and
  - Database approach.
- Both approaches start from the supposition that UoD functions (like HR, Acc, Sales, Payroll,…) are supported by an application, containing a number of programs
- According to the traditional approach, each application contains all necessary data, as its own, organized into files
  - If two applications need the same data, files may be duplicated
    - Duplication leads to data redundancy and inconsistency
  - If files are not duplicated, then the two applications suffer from access concurrency and may have different data access mode needs
- In the database approach all applications use a common database (no application owns the database)

Database Approach



Data Independence

- In the traditional (file) approach to data organization, information about a file’s physical structure was embedded into programs
- The same was true for the first (pre-relational) database systems
- Changes of the file structure induced changes in programs
- This phenomenon was called (program -) data dependence
- The use of declarative manipulation languages relived databases from the program-data dependence

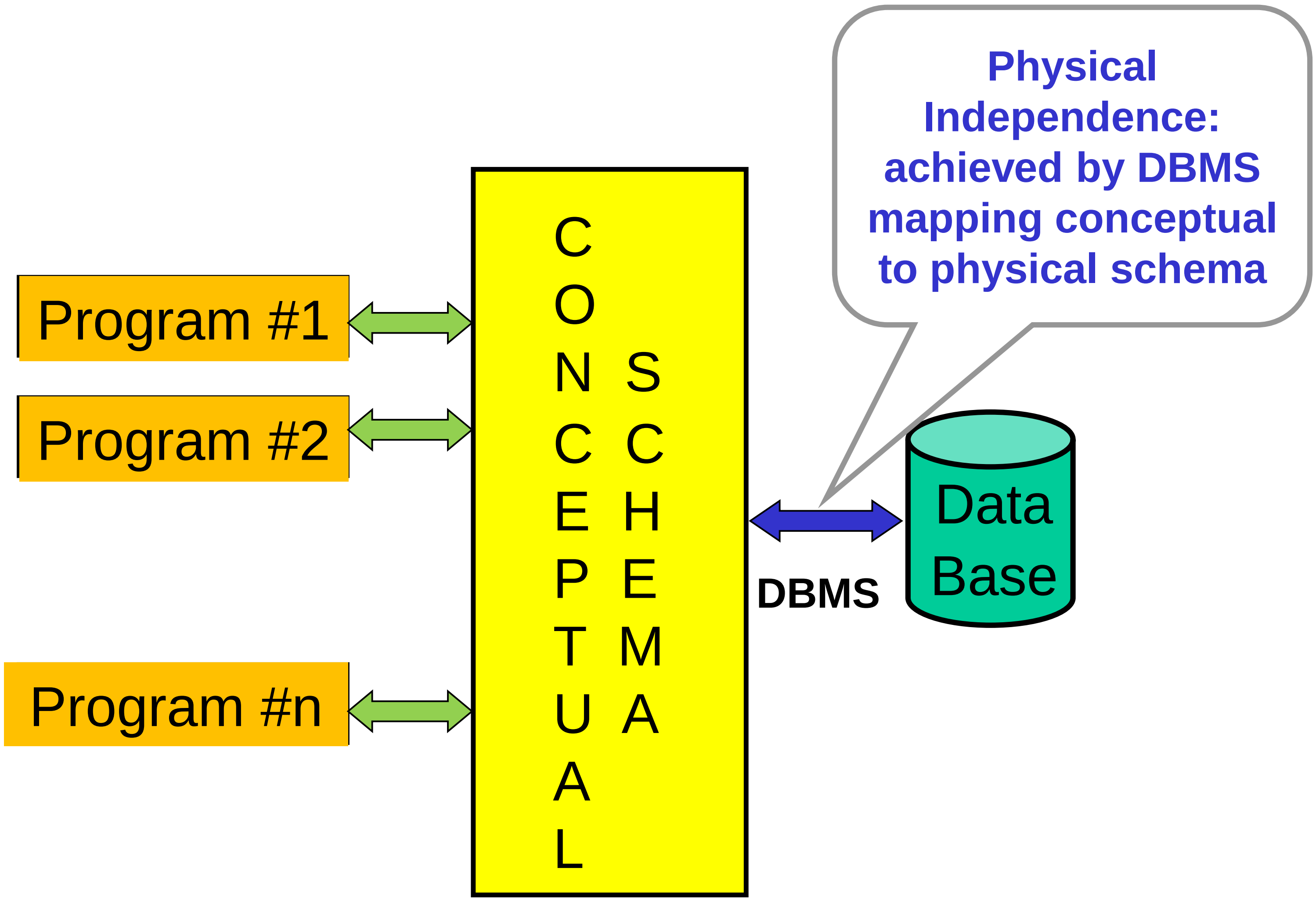
The Three Schema Architecture

- ANSI/SPARK 1975
- Three levels of schema definition (as design categories):
  - External (user view – often EER data model, can also be relational),
  - Conceptual (global view – often EER data model, can also be relational), and
  - Internal (database physical structure – indexes, hashing algorithms, pointers, disk extents,…)
- If the conceptual schema is defined in the EER data model, then we should also talk about implementation (relational) level

Physical Data Independence

- Means: The physical organization of the data is almost independent from the conceptual / logical organization
- Changes to physical schema have no implications on the logical / conceptual layer
- Allow to abstract from the realization of the DBMS storage organization
- Allow to reason about data without worrying about the physical realization
- Allow physical optimization / tuning
- Achieved through conceptual to physical schema mapping performed by DBMS

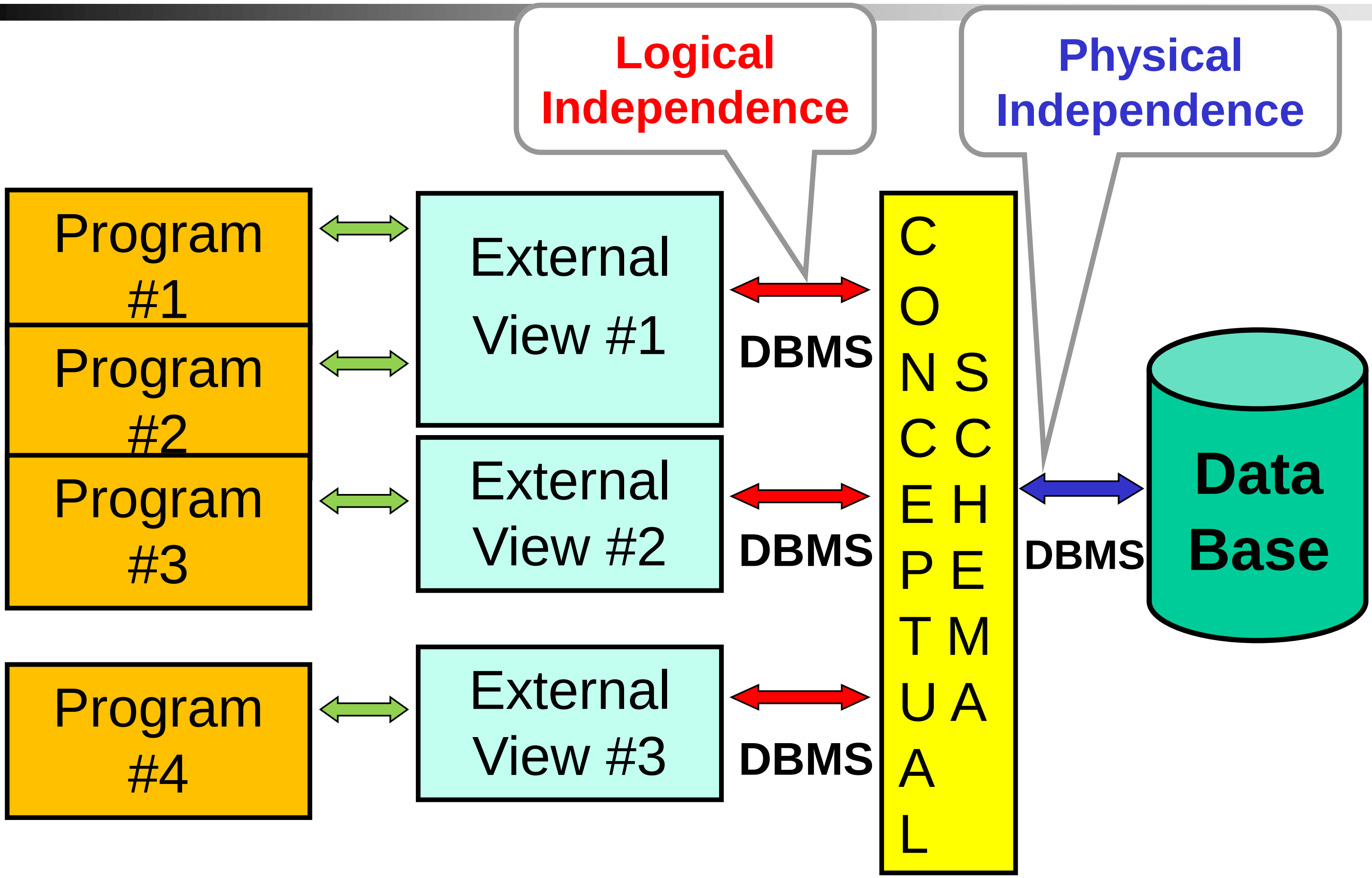
Physical Data Independence



Logical Data Independence

- Means: the external presentation and manipulation of the data is almost independent from the conceptual/logical organization
- Changes to the conceptual/logical schema should not affect the external schema
- Only mapping shall be changed
- An application program should only see the external schema
  - View update problem is solved using virtual views

Logical Data Independence



Summary

- A data model is a (mathematical) abstraction that has structural and dynamic components
  - The structural component is used to represent UoD structure and rules of behavior (constraints between data and relationships)
  - Operations are used to model dynamics of a UoD
- The representation of a UoD by structure and operations constitutes a database design
- Database schema – abstract description of a UoD structure and rules of behavior
- A schema instance – an image of a UoD state – the database itself
- Main advantages of the database over traditional approach:
  - Program – data independence,
  - Data consistency monitored by a DBMS,
  - DBMS controlled data sharing and recovery

Plan for the Next Lecture

- Introduction to the relational data model – motives and basic ideas
- Basic terms and concepts of the relational data model
- Relational schemas and instances
- Constraints of the relational data model

  - *Reading Chapter 5 of the textbook*
    - *sections 5.1 and 5.2*