Database System Concepts and Architecture

Data Models

- **Data Model:**
- A set of concepts to describe the ***structure*** of a database,
- the ***operations*** for manipulating these structures, and certain ***constraints*** that the database should obey.

Data Model Structure and Constraints:

- Constructs are used to define the database structure
- Constructs typically include ***elements*** (and their ***data types***) as well as groups of elements (e.g. ***entity, record, table***), and ***relationships*** among such groups
- Constraints specify some restrictions on valid data; these constraints must be enforced at all times

- **Data Model Operations:**
- These operations are used for specifying database
- *retrievals* and *updates* by referring to the constructs of the data model.
- Operations on the data model may include ***basic model operations*** (e.g. generic insert, delete, update) and ***user-defined operations*** (e.g. compute_student_gpa, update_inventory)

Importance of Data Models

- Are a communication tool
- Give an overall view of the database
- Organize data for various users
- Are an abstraction for the creation of good database

Data Model Basic Building Blocks

- **Entity**: Unique and distinct object used to collect and store data
  - e.g. Student, Car, Sales, City, Rain
- **Attribute**: Characteristic of an entity
  - e.g. Student_ID, Car_Model, Sales_Amount
- **Relationship**: Describes an association among entities
  - **One-to-many (1:M)**
  - **Many-to-many (M:N or M:M)**
  - **One-to-one (1:1)**
- **Constraint**: Set of rules to ensure data integrity

Schemas versus Instances

- **Database Schema:**
- The ***description*** of a database.
- Includes descriptions of the database structure, data types, and the constraints on the database.
- **Schema Diagram:**
- An ***illustrative*** display of (most aspects of) a database schema.
- **Schema Construct:**
- A ***component*** of the schema or an object within the schema, e.g., STUDENT, COURSE.

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Schema Diagram

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

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

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Database

Schemas versus Instances

- Database State:
- The actual data stored in a database at a ***particular moment in time***. This includes the collection of all the data in the database.
- Also called database instance (or occurrence or snapshot).
    - The term *instance* is also applied to individual database components, e.g. *record instance, table instance, entity instance*

Database Schema vs. Database State

- Database State:
- Refers to the ***content*** of a database at a moment in time.
- Initial Database State:
- Refers to the database state when it is initially loaded into the system.
- Valid State:
- A state that satisfies the structure and constraints of the database.

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

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

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

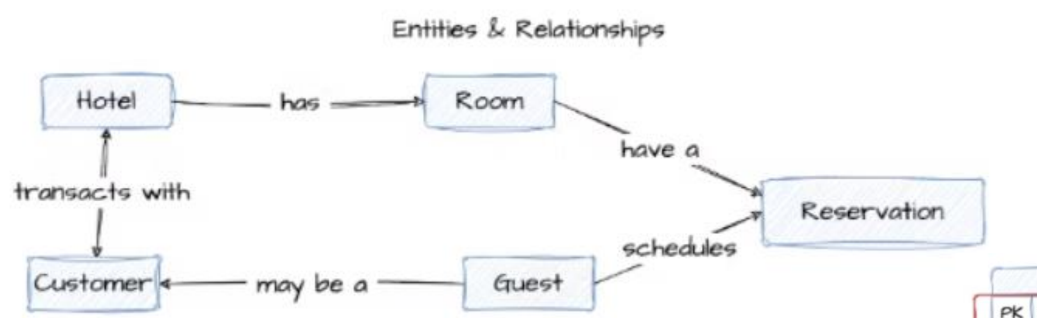
Database State

Database Schema vs. Database State

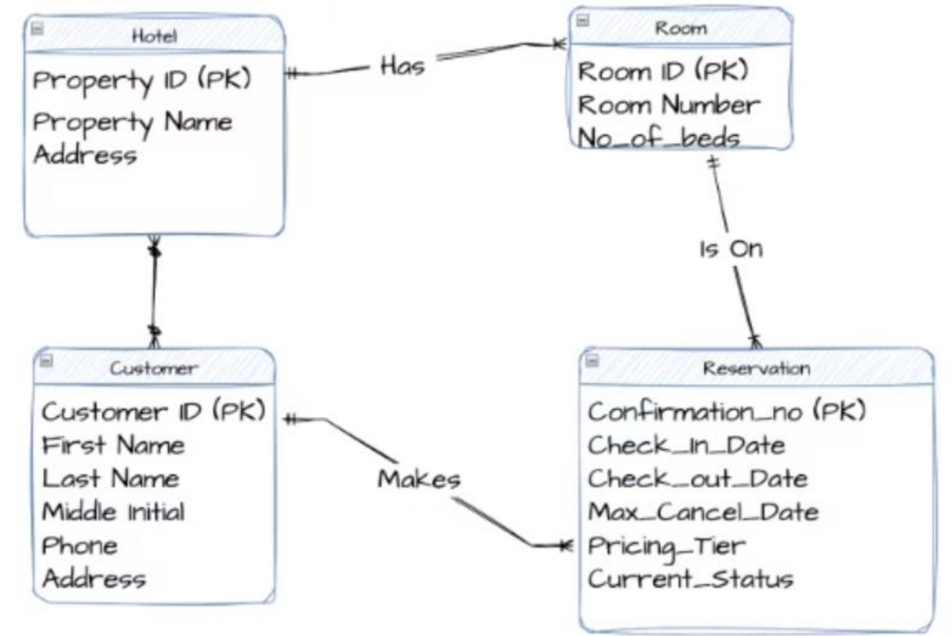
- Distinction
- The ***database schema*** changes very infrequently.
- The ***database state*** changes every time the database is updated.
- **Schema** is also called **intension**.
- **State** is also called **extension**.

Categories of Data Models

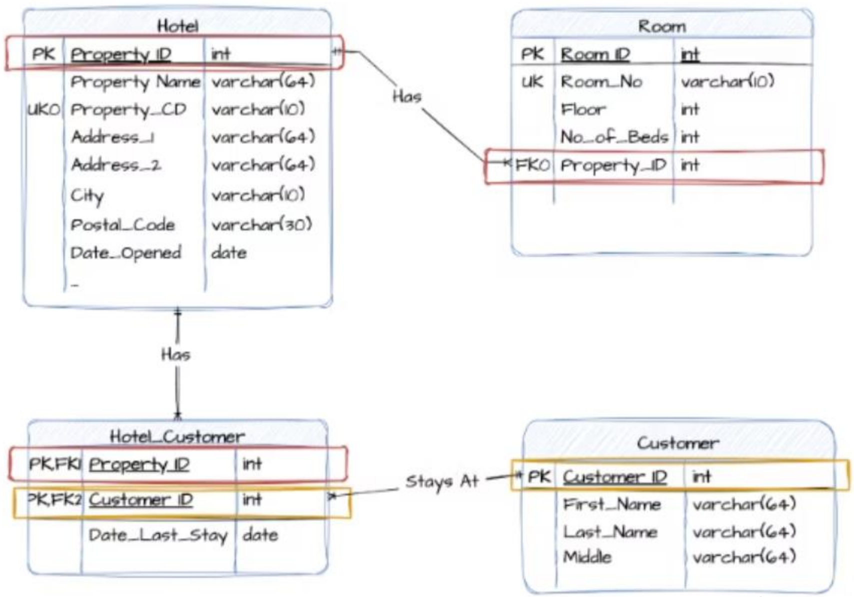
- **Conceptual (high-level, semantic) data models:**
  - Provide concepts that are close to the way many users perceive data.
    - (Also called ***entity-based*** or ***object-based*** data models.)
- **Physical (low-level, internal) data models:**
  - Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals
- **Implementation (representational) data models:**
  - Provide concepts that fall between the above two, used by many commercial DBMS implémentations (e.g. relationnel data model used in many commercial systems).
- **Self-Describing Data Models:**
  - Combine the description of data with the data values. Examples include XML, key-value stores and some NOSQL systems.



Conceptual Data Model



Implementational Data Model



Physical Data Model

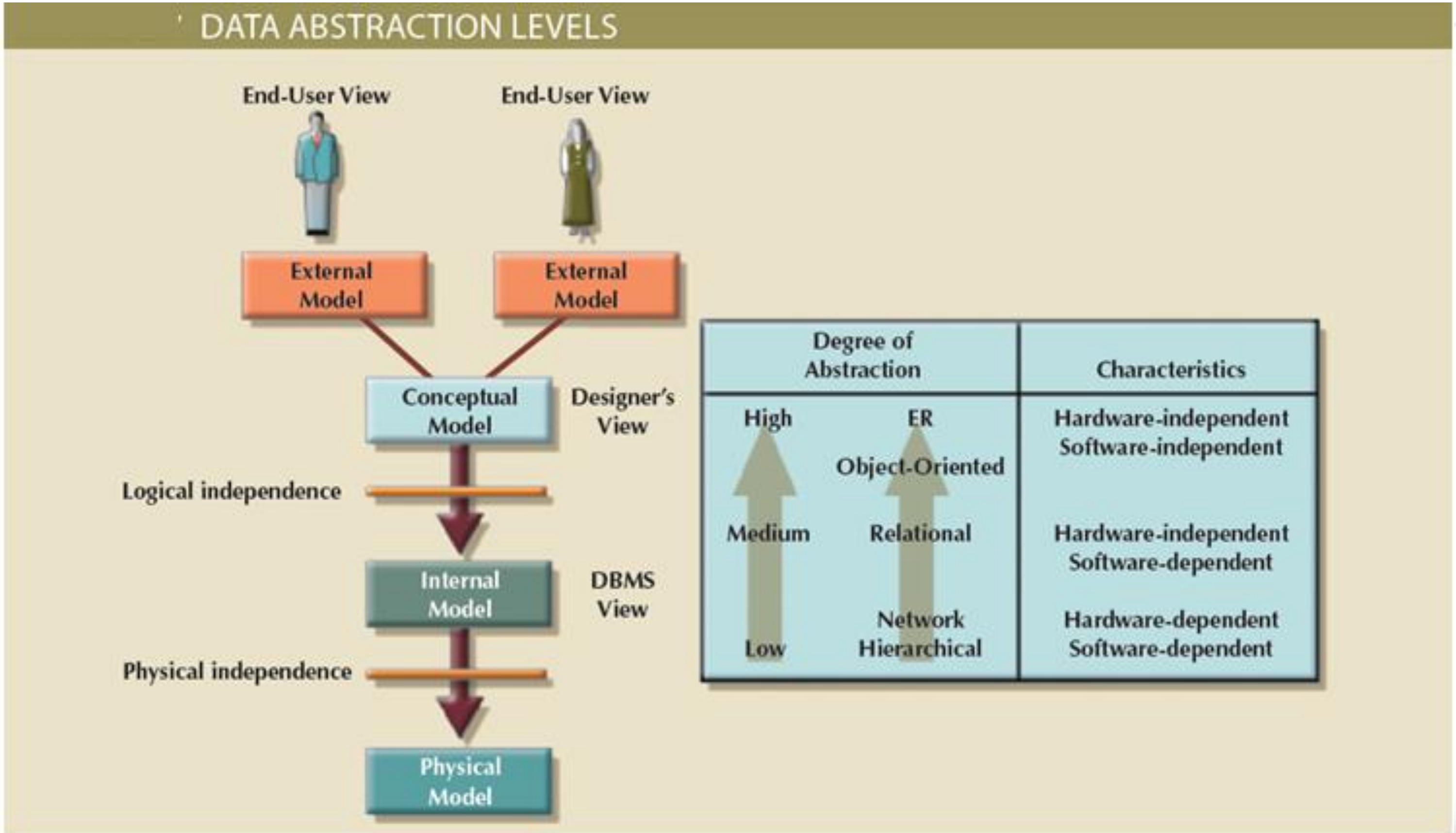
Three-Schema Architecture

- Proposed to support DBMS characteristics of:
  - **Program-data independence.**
  - Support of **multiple views** of the data.

Three-Schema Architecture

- Defines DBMS schemas at ***three*** levels:
  - **Internal schema** at the internal level to describe physical storage structures and access paths (e.g. indexes).
    - Typically uses a **physical** data model.
  - **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.
  - **External schemas** at the external level to describe the various user views.
    - Usually uses the same data model as the conceptual schema.

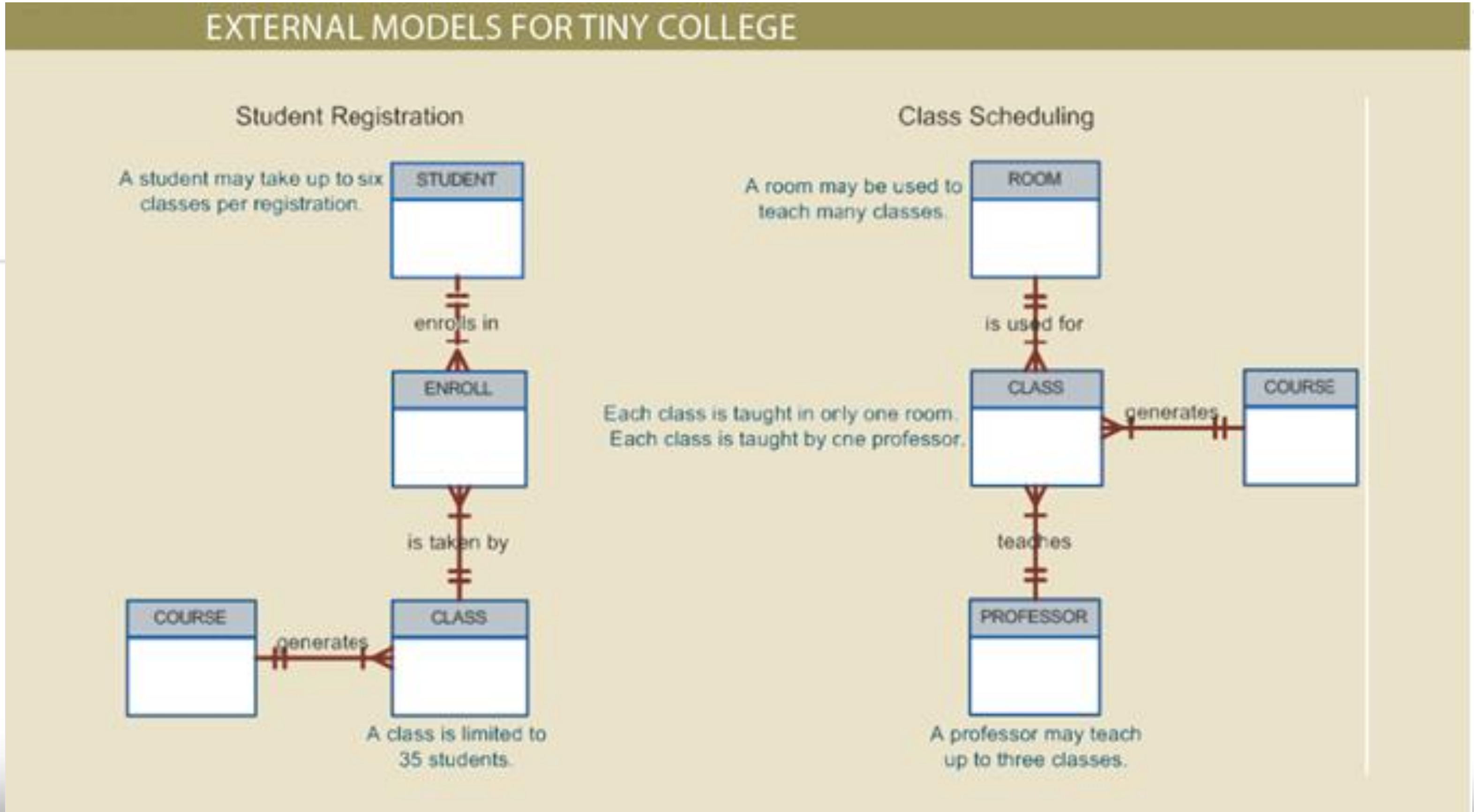
Data Abstraction Levels



The External Model

- End users’ view of the data environment
- ER diagrams are used to represent the external views
- **External schema**: Specific representation of an external view

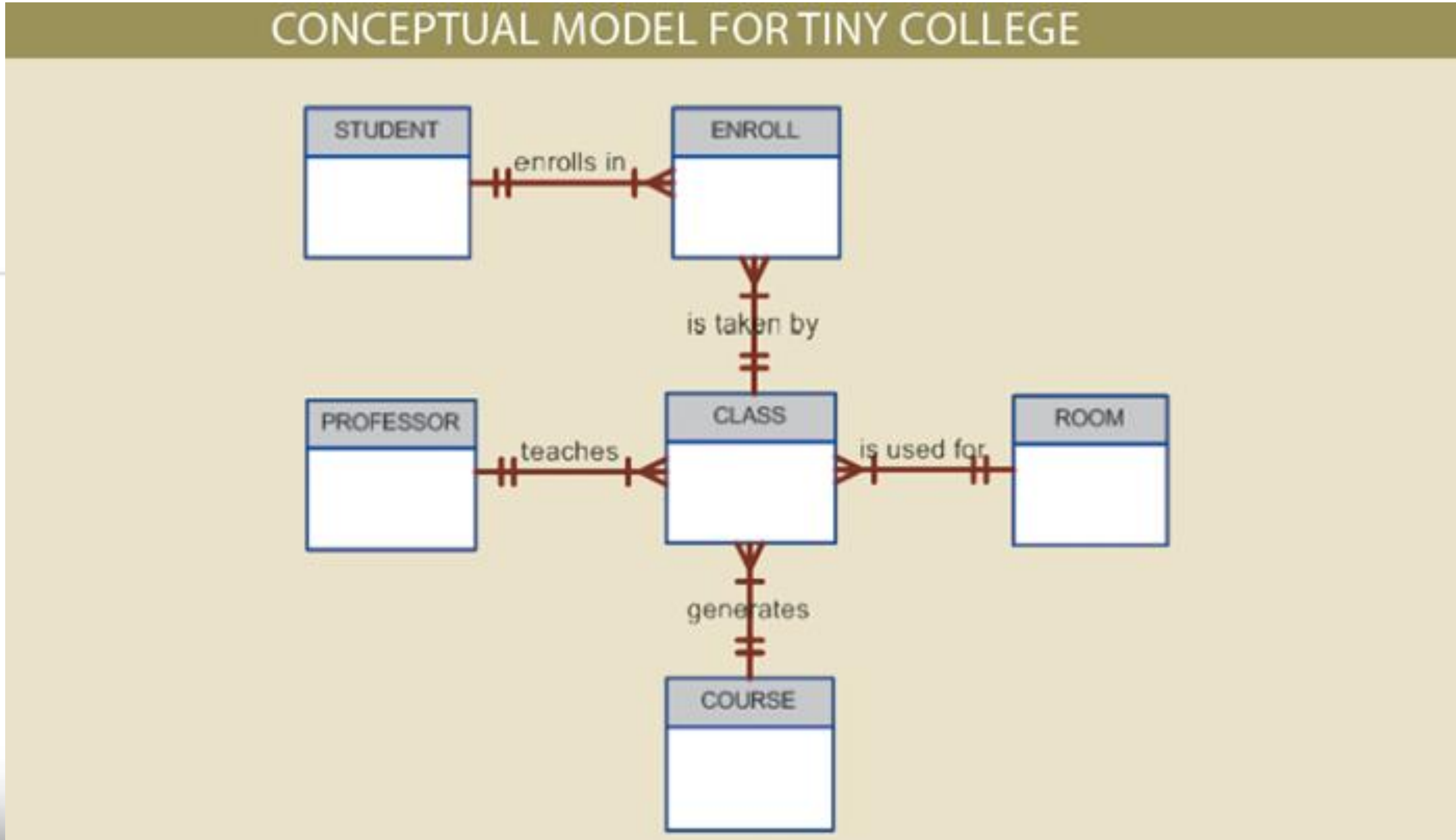
External Models For Tiny College



The Conceptual Model

- Represents a global view of the entire database by the entire organization
- **Conceptual schema**: Basis for the identification and high-level description of the main data objects
- Has a macro-level view of data environment
- Is software and hardware independent
- **Logical design**: Task of creating a conceptual data model

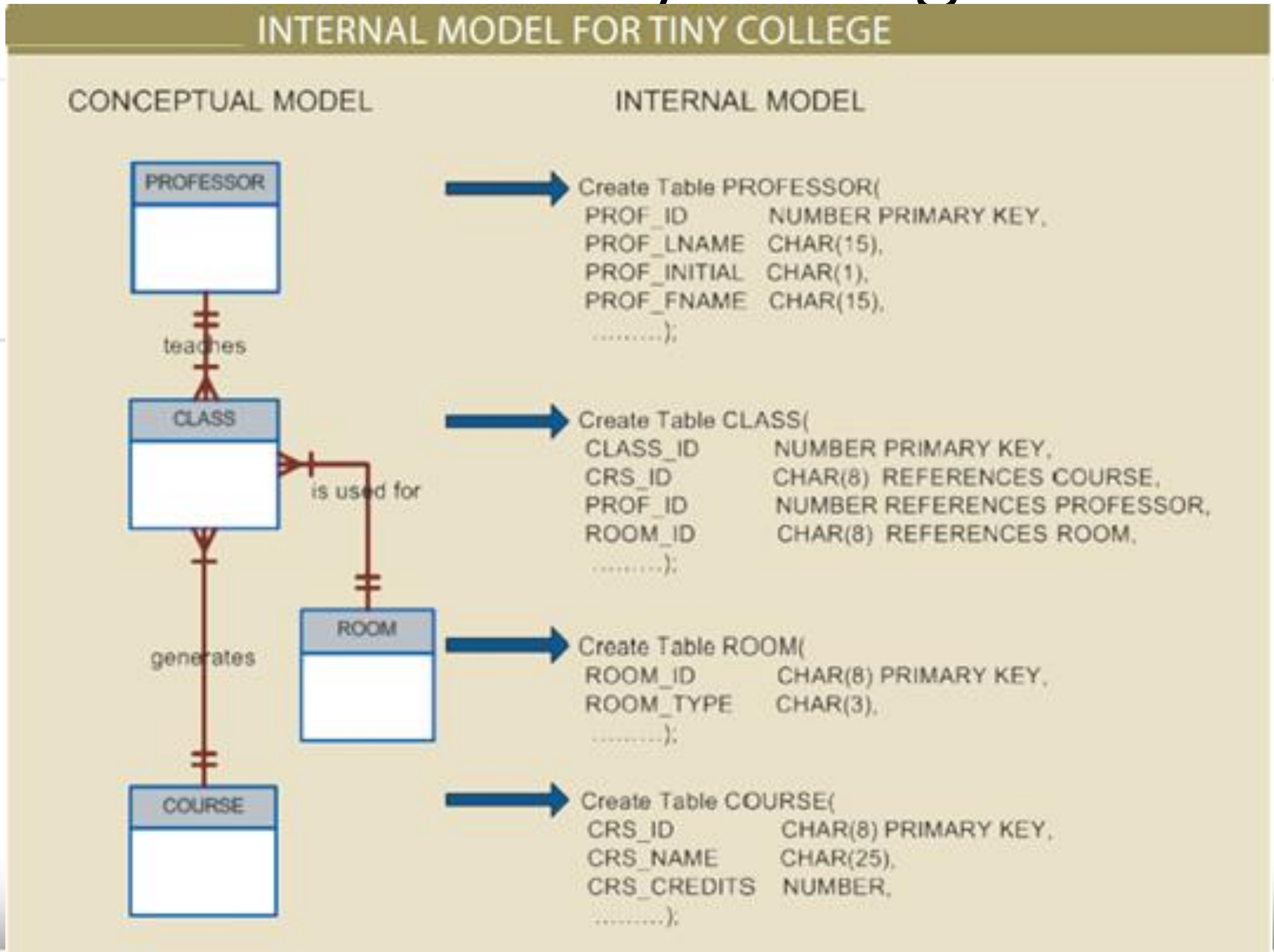
Conceptual Model For Tiny College



The Internal Model

- Representing database as seen by the DBMS mapping conceptual model to the DBMS
- **Internal schema**: Specific representation of an internal model
  - Uses the database constructs supported by the chosen database
- Is software dependent and hardware independent
- **Logical independence**: Changing internal model without affecting the conceptual model

Internal Model for Tiny College



The Physical Model

- Operates at lowest level of abstraction
- Describes the way data are saved on storage media such as disks or tapes
- Requires the definition of physical storage and data access methods
- Relational model aimed at logical level
  - Does not require physical-level details
- **Physical independence**: Changes in physical model do not affect internal model

Levels of Data Abstraction

LEVELS OF DATA ABSTRACTION

MODEL	DEGREE OF ABSTRACTION	FOCUS	INDEPENDENT OF
External	High	End-user views	Hardware and software
Conceptual		Global view of data (database model independent)	Hardware and software
Internal		Specific database model	Hardware
Physical	Low	Storage and access methods	Neither hardware nor software

Three-Schema Architecture

- Mappings among schema levels are needed to transform requests and data.
  - Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.
  - Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g. formatting the results of an SQL query for display in a Web page)

Data Independence

Logical Data Independence

- Is the capacity to change the conceptual schema without having to change external schemas or application programs.
- It is difficult to achieve logical data independence.
- We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item.

Physical Data Independence

- is the capacity to change the internal schema without having to change the conceptual schema.
- It is easier to achieve logical data independence.
- Changes to the internal schema may be needed because some physical files were reorganized— by creating additional access structures—to improve the performance of retrieval or update.
- For E.g. *List of all sections offered in fall 2008 will not be changed* but will executed **more efficient** by utilizing the **new access path**

Logical Data Independence

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

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
18	Smith	135	CS3380	A

DBMS Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
  - High-Level or Non-procedural Languages
  - Low Level or Procedural Languages

DBMS Languages

- Data Definition Language (DDL)
  - Used by the DBA and database designers to specify the conceptual schema of a database.
  - Used to define or modify the structure of database objects such as tables, schemas, indexes, etc.
  - In many DBMSs, the DDL is also used to define internal and external schemas (views).
  - In some DBMSs, separate **storage definition language (SDL)** and **view definition language (VDL)** are used to define internal and external schemas.
    - SDL is typically realized via DBMS commands provided to the DBA and database designers

DBMS Languages

- **Data Manipulation Language (DML):**
  - **Used to manipulate the data within the database (e.g., inserting, updating, deleting, or retrieving data).**
  - Used to specify database retrievals and updates
  - DML commands (data sublanguage) can be *embedded* in a general-purpose programming language (host language), such as COBOL, C, C++, or Java.
    - A library of functions can also be provided to access the DBMS from a programming language
  - Alternatively, stand-alone DML commands can be applied directly (called a *query language*).

DBMS Languages

DDL Commands

CREATE: Creates a new database object (e.g., table, index).
ALTER: Modifies an existing database object (e.g., adding a column to a table).
DROP: Deletes an existing database object.
TRUNCATE: Removes all records from a table, resetting its structure.

DML Commands

SELECT: Retrieves data from the database.
INSERT: Adds new data to a table.
UPDATE: Modifies existing data.
DELETE: Removes data from a table.

Types of DML

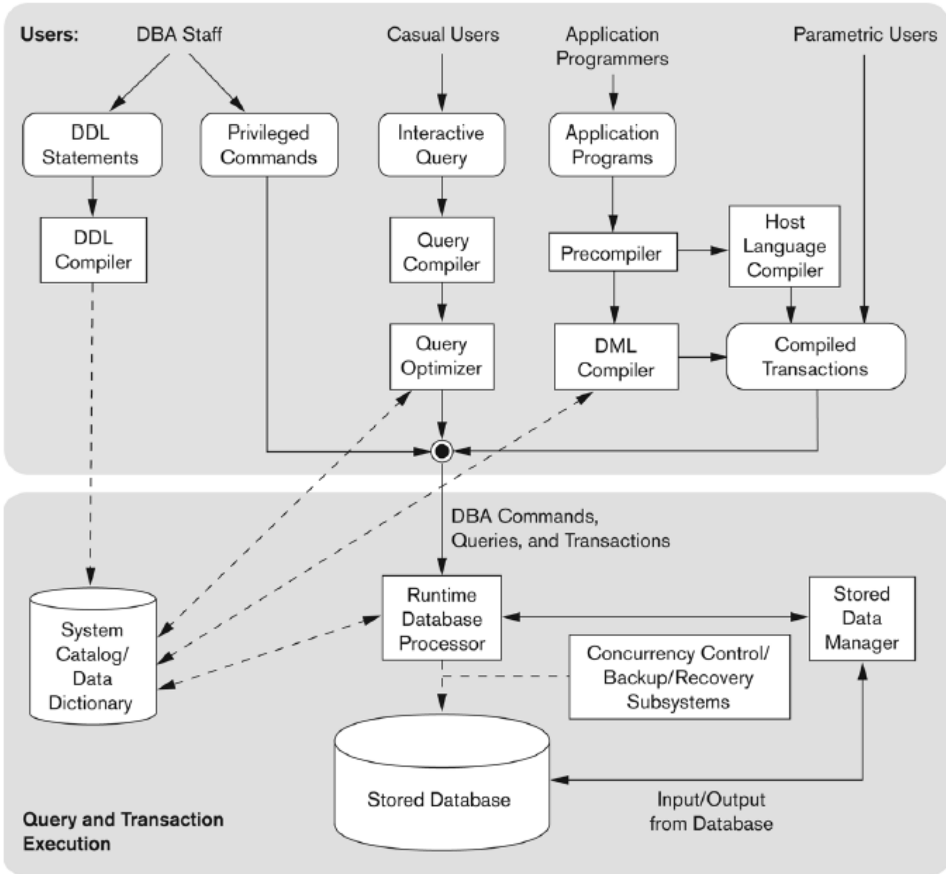
High-Level or Non-procedural Languages

- These languages focus more on the "what" rather than the "how." The programmer specifies **what** needs to be done without dictating the exact sequence of operations.
- These can specify and retrieve many records in a single DML called as set at a time
- Also called **declarative** languages.
- Examples: SQL, HTML, Prolog.

Low Level or Procedural Languages

- These languages require the programmer to explicitly define **how** to perform tasks. They provide direct control over the system's resources, often requiring a step-by-step description of procedures.
- It requires language construct such as looping to retrieve each record, so that’s why it is called as record at a time
- Examples: C, Assembly language, Fortran.

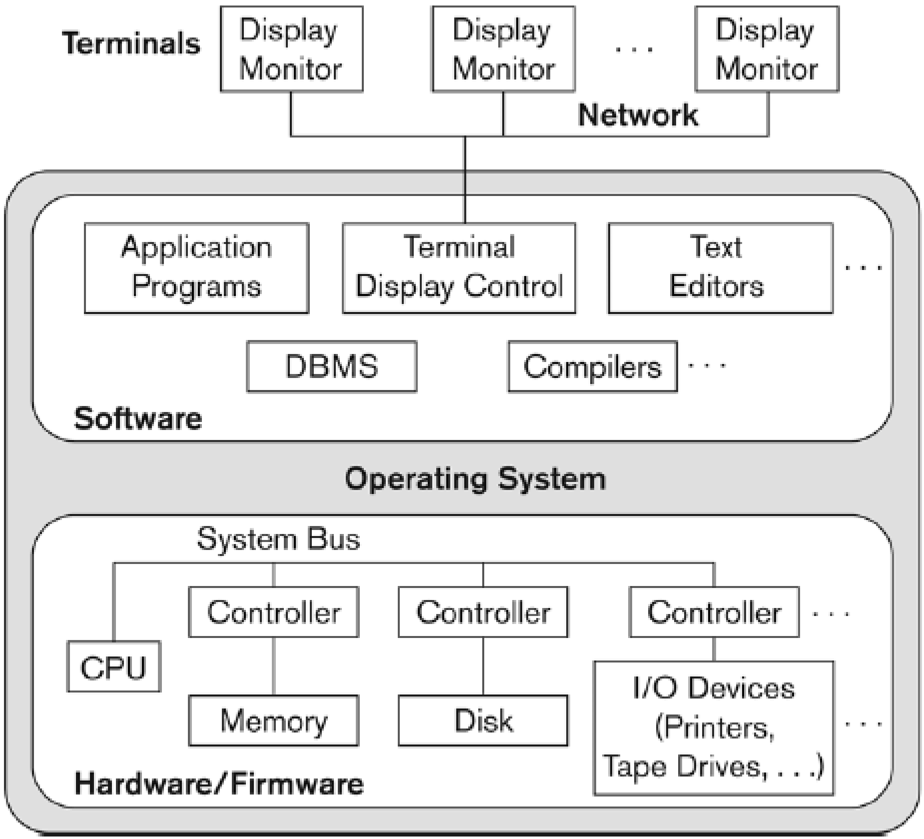
Typical DBMS Component Modules



Centralized and Client-Server DBMS Architectures

- Centralized DBMS:
  - Combines everything into single system including- DBMS software, hardware, application programs, and user interface processing software.
  - User can still connect through a remote terminal – however, all processing is done at centralized site.

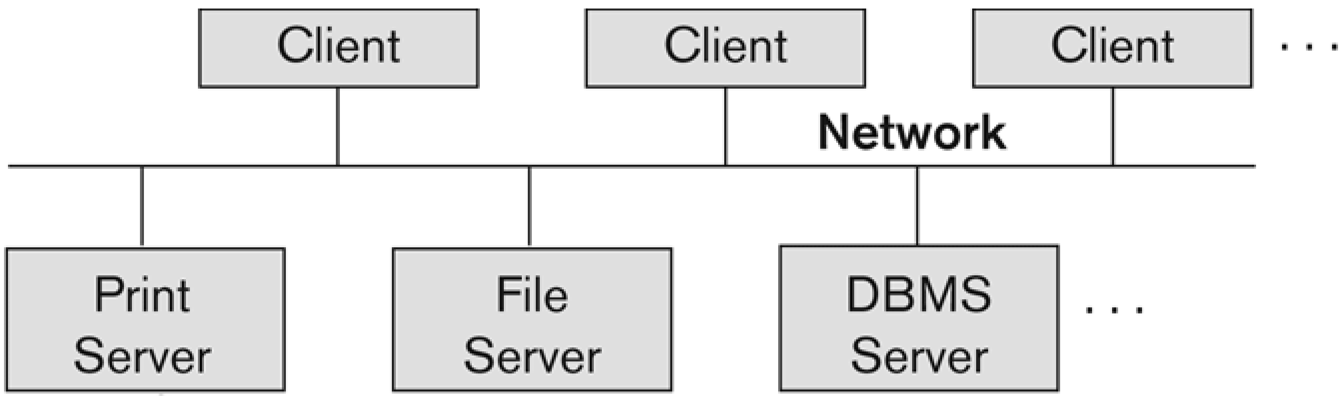
A Physical Centralized Architecture



Basic 2-tier Client-Server Architectures

- Specialized Servers with Specialized functions
  - Print server
  - File server
  - DBMS server
  - Web server
  - Email server
- Clients can access the specialized servers as Needed

Logical two-tier client server architecture



Clients

- Provide appropriate interfaces through a client software module to access and utilize the various server resources.
- Clients may be 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.)

DBMS Server

- Provides database query and transaction services to the clients
- Relational DBMS servers are often called SQL servers, query servers, or transaction servers
- Applications running on clients utilize an Application Program Interface (**API**) to access server databases via standard interface such as:
  - ODBC: Open Database Connectivity standard
  - JDBC: for Java programming access

Two Tier Client-Server Architecture

- Client and server must install appropriate client module and server module software for ODBC or JDBC
- A client program may connect to several DBMSs, sometimes called the data sources.
- In general, data sources can be files or other non-DBMS software that manages data.

Three Tier Client-Server Architecture

Intermediate Layer called Application Server or Web Server

- Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server
- Acts like a conduit for sending partially processed data between the database server and the client.

Three-tier Architecture Can Enhance Security

- Database server only accessible via middle tier
- Clients cannot directly access database server
- Clients contain user interfaces and Web browsers
- The client is typically a PC or a mobile device connected to the Web

Three-tier client-server architecture