4CCS1DBS

Database Systems

4CCS1DBS
Database Systems

- Lecturers:
  - Dr Vasa Curcin
  - Dr Sophia Tsoka
- TAs:
  - Elmira Amiri Souri, Mohamad Alawie, Sara Boutamina, Federica de Chiara, Zhuoling Huang, Jenjira Jaimunk, Paul Nuller, Nicolas Volken, Andreas Xydis

- Textbook ‘Database Systems’ R. Elmasri, S. Navathe

- Slides available from KEATS
  - https://keats.kcl.ac.uk/course/view.php?id=66951

4CCS1DBS
Database Systems

- Lecturer contact information
  - Dr Vasa Curcin
    - vasa.curcin@kcl.ac.uk
    - BH (N) 7.05
    - office hours Monday 13:00 – 15:00
  - Dr Sophia Tsoka
    - sophia.tsoka@kcl.ac.uk
    - BH (N) 5.19
    - office hours Wednesday 11:00 – 13:00

4CCS1DBS
Database Systems

- How to contact us about any questions or help with Module
  - Discussion Group on KEATS
  - Help Desk
  - Office Hours

4CCS1DBS
Database Systems

- Lectures
    - Monday 3-5pm
- Large Group Tutorials
    - Monday 5-6pm
- Lab Practicals (5 sessions)
    - Revision required
- Assessment
    - Exam (85%)
    - Coursework (15%)

4CCS1DBS
Database Systems

- Lab Practicals
  - Every 2 weeks
  - Schedule slot is shared with Data Structures - DST
- Covers the material in the module with aid of TAs
- Prepares you for doing the Coursework
- Attendance is required and is monitored

4CCS1DBS
Database Systems

- Major Coursework (15% of final mark)
  - Practice Designing a Database from Scratch
  - Implementing the Database in SQL
  - Technical Details (i.e. version of SQL) will be provided in the CW description

- Deadlines for the coursework will be confirmed on KEATS

4CCS1DBS
Database Systems

- Lectures, Tutorials and Labs
  - Revision required regularly
- Revising
  - Slides and textbook
- Contacting the lecturer and TAs
  - Office hours better than email

List of Lectures

Lecture	Topic	Lecturer
1	Introduction to Database Systems	ST
2	Data Modeling Using the Entity-Relationship Model	ST
3	Relational Data Model	VC
4	SQL 1	VC
5	SQL 2	VC
6	Relational Algebra 1	VC
7	Relational Algebra 2	VC
8	Advanced Topics and Security Issues	ST
9	Normalisation 1	ST
10	Normalisation 2	ST

4CCS1DBS Overview – Learning Objectives

- An introduction to main concepts in the design and implementation of database systems
  - Entity-Relationship Modelling
  - The Relational Data Model and Relational Database Constraints
  - Structured Query Language (SQL) - schema definition, constraints and queries
  - Relational Algebra
  - Functional Dependencies and Normalisation for Relational Databases

Lecture 1 – Learning Outcomes

- An introduction to databases
- Understand the use of database systems
- Terminology, main definitions
- Database architectures, data models

Part 1 – Databases and Database Use

- Types of Databases and Database Applications
- Basic Definitions
- Typical DBMS Functionality
- Example of a Database (UNIVERSITY)
- Main Characteristics of the Database Approach
- Advantages of Using the Database Approach
- When Not to Use Databases

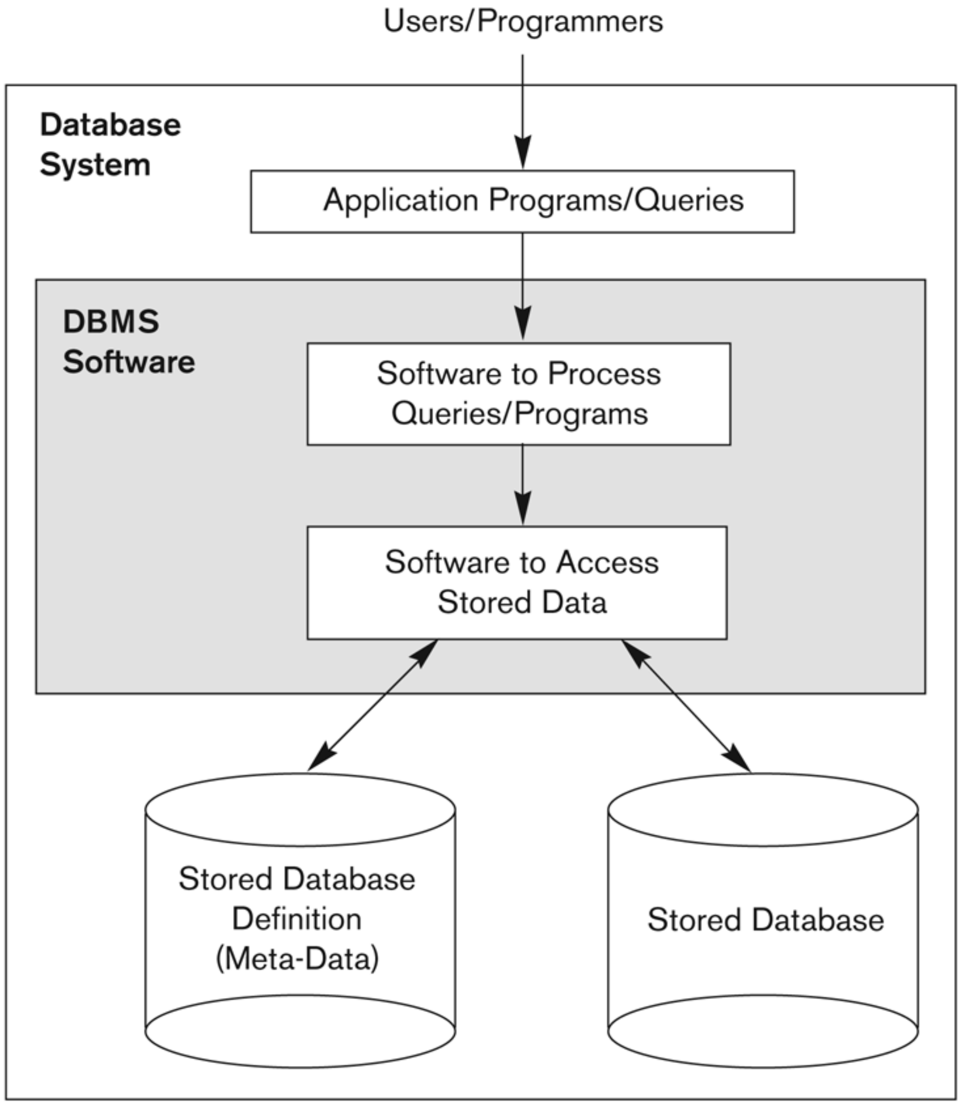
Types of Databases and Database Applications

- Traditional Applications:
  - Numeric and Textual Databases
- More Recent Applications:
  - Multimedia Databases
  - Geographic Information Systems (GIS)
  - Data Warehouses
  - Real-time and Active Databases
  - Many other applications

Basic Definitions

- **Database:**
  - organised
  - A collection of related data.
- **Data:**
  - Known facts that can be recorded and have an implicit meaning.
- **Mini-world:**
  - part of real world related to the database
  - Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
- **Database Management System (DBMS):**
  - A software package/system to facilitate the creation and maintenance of a computerised database. maintaining, updating, queries…
- **Database System:**
  - The DBMS software together with the data itself. Sometimes the applications are also included.

Simplified Database System Environment



Typical DBMS Functionality

- *Define* a particular database in terms of its data types, structures, and constraints
- *Construct* or *load* the initial database contents on a secondary storage medium
- *Manipulating* the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing the database through Web applications
- *Processing* and *Sharing* by a set of concurrent users and application programs – yet, keeping all data valid and consistent

Typical DBMS Functionality

- Other features:
  - Protection or Security measures to prevent unauthorised access
  - “Active” processing to take internal actions on data
  - Presentation and Visualisation of data
  - Maintaining the database and associated programs over the lifetime of the database application
    - Called database, software, and system maintenance

Example: Database with Conceptual Data Model

- **Mini-world for the example:**
  - Part of a UNIVERSITY environment.
- **Some mini-world *entities*:**
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (academic) DEPARTMENTs
  - INSTRUCTORs

Example: Database with Conceptual Data Model

- **Some mini-world *relationships*:**
  - SECTIONs *are of specific* COURSEs
  - STUDENTs *take* SECTIONs
  - COURSEs *have prerequisite* COURSEs
  - INSTRUCTORs *teach* SECTIONs
  - COURSEs *are offered by* DEPARTMENTs
  - STUDENTs *major in* DEPARTMENTs

- Note: The above entities and relationships are typically expressed in a conceptual data model, such as the ENTITY-RELATIONSHIP data model (discussed next week)

Example of a Simple 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

Main Characteristics of the Database Approach

- **Self-describing nature of a database system:**
  - A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints)
  - The description is called **meta-data**.
  - This allows the DBMS software to work with different database applications.
- **Insulation between programs and data:**
  - Called **program-data independence**.
  - Allows changing data structures and storage organisation without having to change the DBMS access programs.

Example of a Simplified Database Catalog

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

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Main Characteristics of the Database Approach

- **Data Abstraction:**
  - A **data model** is used to hide storage details and present the users with a conceptual view of the database.
  - Programs refer to the data model constructs rather than data storage details
- **Support of multiple views of the data:**
  - Each user may see a different view of the database, which describes **only** the data of interest to that user.

Main Characteristics of the Database Approach (cont.)

- **Sharing of data and multi-user transaction processing:**
  - Allowing a set of **concurrent users** to retrieve from and to update the database.
  - *Concurrency control* within the DBMS guarantees that each **transaction** is correctly executed or aborted
  - *Recovery* subsystem ensures each completed transaction has its effect permanently recorded in the database
  - **OLTP** (Online Transaction Processing) is a major part of database applications. This allows hundreds of concurrent transactions to execute per second.

Advantages of Using the Database Approach

불필요한 중복

- Controlling redundancy in *data storage* and in *development* and *maintenance* efforts.
- Providing multiple interfaces to different classes of users, facilitate sharing data across users.
- Restricting unauthorised access to data.
- Providing Storage Structures (e.g. indexes) for efficient query processing.
- Providing backup and recovery services.
- Representing complex relationships among data.
- Enforcing integrity constraints on the database.
- Drawing inferences and actions from the stored data using deductive and active rules

Additional Implications of Using the Database Approach

- Potential for enforcing standards:
  - This is very crucial for the success of database applications in large organisations. **Standards** refer to data item names, display formats, screens, report structures, meta-data (description of data), Web page layouts, etc.
- Reduced application development time:
  - Incremental time to add each new application is reduced.

Additional Implications of Using the Database Approach (continued)

- Flexibility to change data structures:
  - Database structure may evolve as new requirements are defined.
- Availability of current information:
  - Extremely important for on-line transaction systems such as airline, hotel, car reservations.
- Economies of scale:
  - Wasteful overlap of resources and personnel can be avoided by consolidating data and applications across departments.

When not to use a DBMS

- Main inhibitors (costs) of using a DBMS:
  - High initial investment and possible need for additional hardware.
  - Overhead for providing generality, security, concurrency control, recovery, and integrity functions.
- When a DBMS may be unnecessary:
  - If the database and applications are simple, well defined, and not expected to change.
  - If there are stringent real-time requirements that may not be met because of DBMS overhead.
  - If access to data by multiple users is not required.

When not to use a DBMS

- When no DBMS may suffice:
  - If the database system is not able to handle the complexity of data because of modeling limitations
  - If the database users need special operations not supported by the DBMS.

Summary of Part 1

- Types of Databases and Database Applications
- Basic Definitions
- Typical DBMS Functionality
- Example of a Database (UNIVERSITY)
- Main Characteristics of the Database Approach
- Advantages of Using the Database Approach
- When Not to Use Databases

Part 2 : Database System Concepts and Architecture

- Data Models and Their Categories
- Schemas, Instances, and States
- Three-Schema Architecture
- DBMS Languages
- Centralised and Client-Server Architectures
- Classification of DBMSs

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

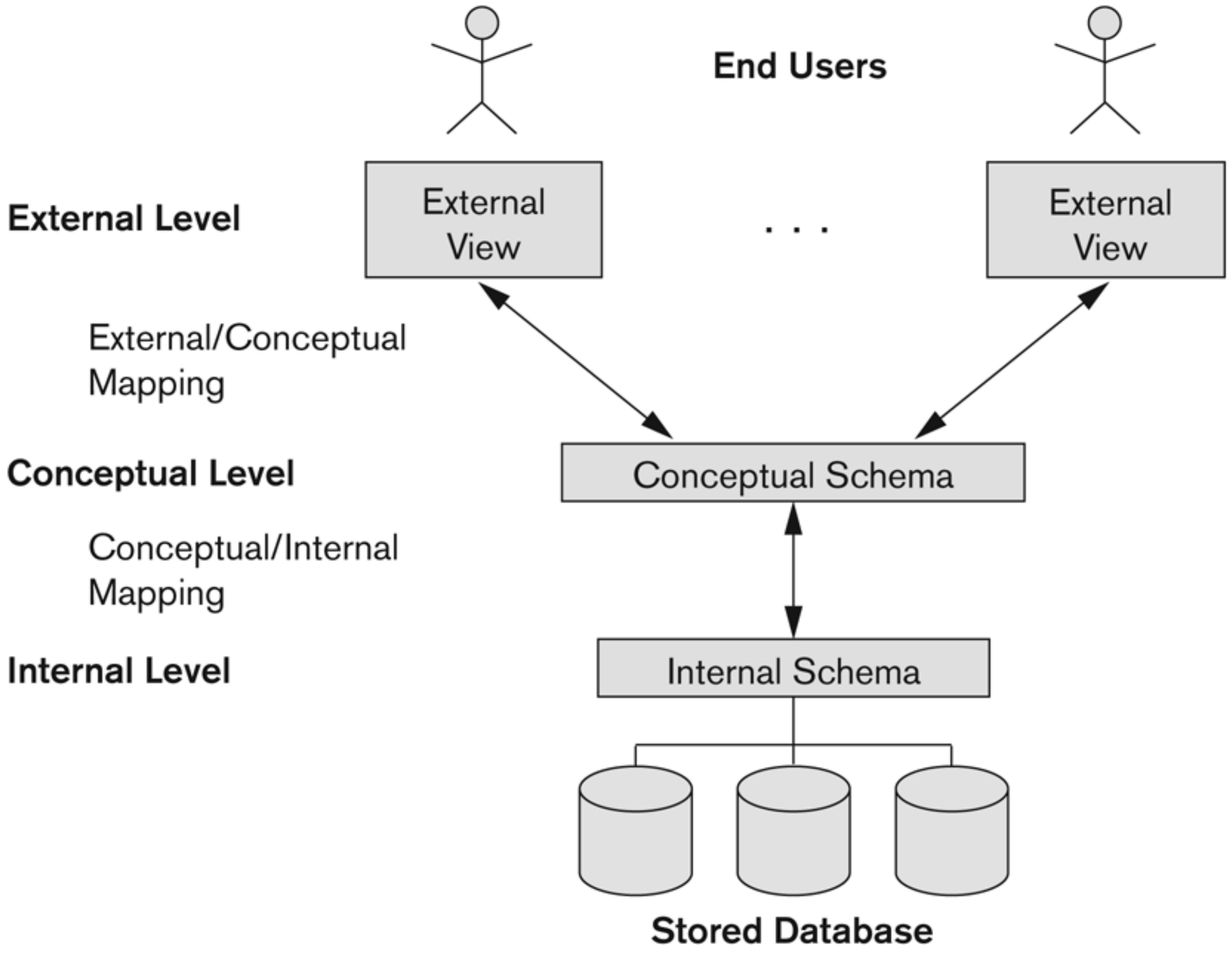
Data Models (continued)

- **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)
    - ***user-defined operations*** (e.g. compute_student_gpa, update_inventory)

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 implementations (e.g. relational data models used in many commercial systems).

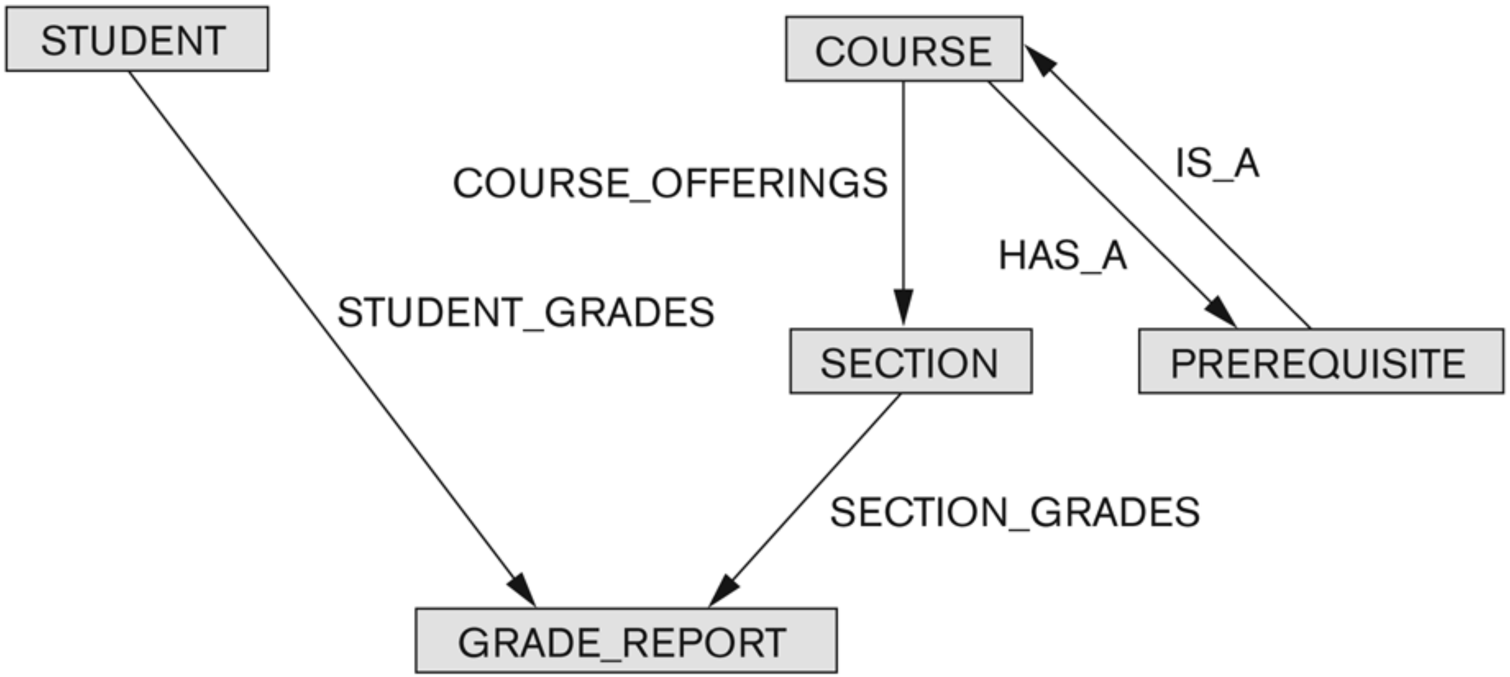
The Three-Schema Architecture



Types of Data Models

- Relational Model
- Network Model
- Hierarchical Model
- Object-oriented Data Models
- Object-Relational Models

Example of Network Model Schema



Network Model

- Advantages:
  - Network Model is able to model complex relationships
  - Can handle most situations for modeling using record types and relationship types.
  - Language is navigational; uses constructs like FIND, FIND member, FIND owner, FIND NEXT within set, GET, etc.
    - Programmers can do optimal navigation through the database.
- Disadvantages:
  - Navigational and procedural nature of processing
  - Database contains a complex array of pointers that thread through a set of records.
    - Little scope for automated “query optimisation”

The Relational Data Model

- **Relational Model:**
  - Proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82.
  - Appears in several commercial products (e.g. DB2, ORACLE, MS SQL Server, SYBASE, INFORMIX).
  - Several free open source implementations, e.g. MySQL, PostgreSQL
  - Currently most dominant for developing database applications
- **The relational data model will be discussed in detail in following lectures of this module**

Database Schema

- 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.

Example of a Database Schema

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

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.

Database *Schema* vs. Database *State*

- Distinction
  - The ***database schema*** changes infrequently.
  - The ***database state*** changes every time the database is updated.

- **Schema** is also called **intension**.
- **State** is also called **extension**.

Example of a Database State

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

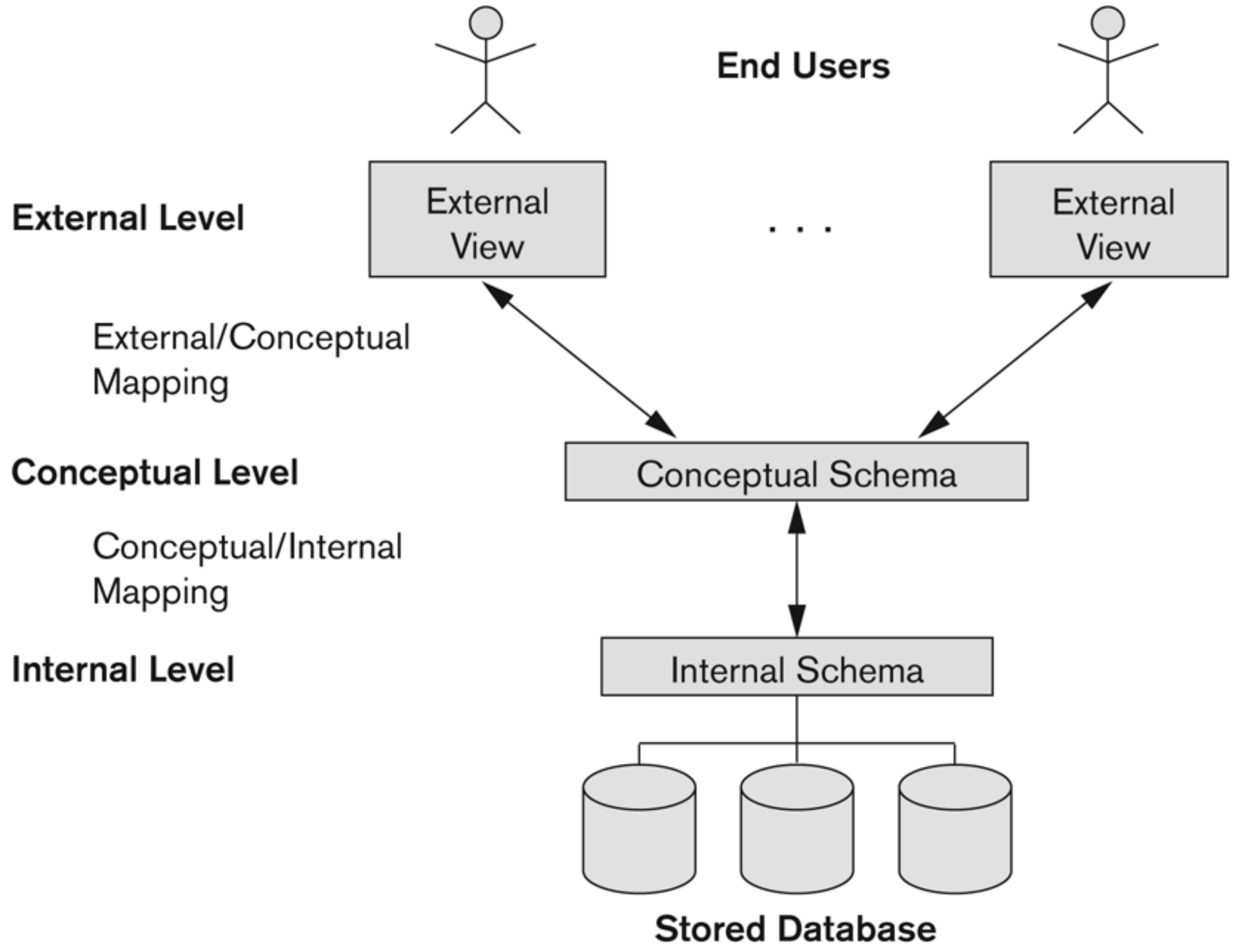
Three-Schema Architecture

- Proposed to support DBMS characteristics of:
  - **Program-data independence.**
  - Support of **multiple views** of the data.
- Not explicitly used in commercial DBMS products, but has been useful in explaining database system organisation

Three-Schema Architecture

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

The Three-Schema Architecture



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)

DBMS Languages

- Data Definition Language (DDL)
  - Specify conceptual and internal schemas
- Data Manipulation Language (DML)
  - Used to query and manipulate the database
    - High-Level or Non-procedural Languages: These include the relational language SQL
      - May be used in a standalone way or may be embedded in a programming language
    - Low Level or Procedural Languages:
      - These must be embedded in a general-purpose programming language

DBMS Languages

- **Data Definition Language (DDL):**
  - Used by the DBA and database designers to *specify* the conceptual schema of a database.
  - 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 realised via DBMS commands provided to the DBA and database designers

DBMS Languages

- **Data Manipulation Language (DML):**
  - Used to specify database *retrievals* and *updates*
  - DML commands (data sublanguage) can be *embedded* in a general-purpose programming language (host language), such as 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*).

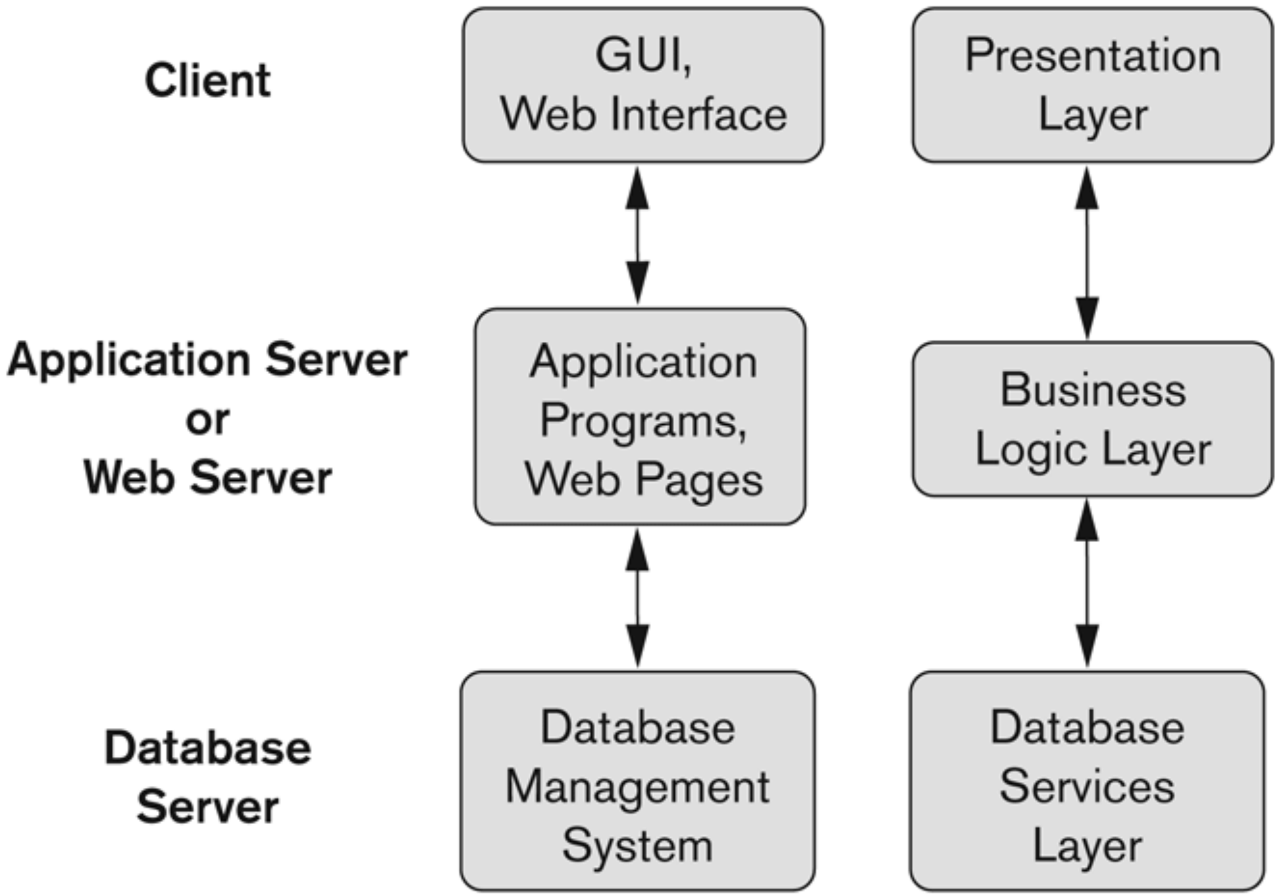
Centralised and Client-Server DBMS Architectures

- Centralised 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 centralised site.

Three Tier Client-Server Architecture

- Common for Web applications
- 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

Three-Tier Client-Server Architecture



Classification of DBMSs

- Based on the data model used
  - Traditional: Relational, Network, Hierarchical.
  - Emerging: Object-oriented, Object-relational.
- Other classifications
  - Single-user (typically used with personal computers) vs. multi-user (most DBMSs).
  - Centralised (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)

Variations of Distributed DBMSs (DDBMSs)

- Homogeneous DDBMS
  - Same DBMS in all sites
- Heterogeneous DDBMS
  - Several autonomous DBMS software at multiple sites, connected through network
- Federated or Multidatabase Systems
  - DBMSs loosely coupled, have some autonomy

Summary of Part 2

- Data Models and Their Categories
- Schemas, Instances, and States
- Three-Schema Architecture
- DBMS Languages
- Centralised and Client-Server Architectures
- Classification of DBMSs