COMP9311 Database Systems

Xuemin Lin
School of Computer Science and Engineering
Office: K17 503
E-mail: lxue@cse.unsw.edu.au
Ext: 6493
http://www.cs.unsw.edu.au/~lxue

WWW home address of 9311:
http://www.cse.unsw.edu.au/~cs9311

Course Information

- Lectures: 18:00 - 21:00 (Mon)
- Location: Central Lecture Block 7 (K-E19-104)
- Lab: week 2 – 12
- Consultation Time: 4:00pm – 5:00pm (Mon)
  - Place: K17-201B.

Course Information(cont)

- 3 assignments, 2 projects, final exam
- Assignments (50%):
  - Ass 1: Data Modelling. Relational/Algebra (10%) (week 2-5)
  - Ass 2: DB design Theory + Transaction + UML (20%) (week 6-9)
  - Ass 3: Graph DB (20%) (week 10-12)
- **Penalty for later submissions: 0 mark will be given to any later submission**.
- Projects (50%)
  - Proj1: 25% (due by week 4-7)
  - Proj2: 25% (due by week 8-11)
- **Penalty for later submissions:** *10% reduction for the 1st day, then 30% reduction*.

Course Information(cont)

- **Exam:** 100%
  - If you are ill on the day of the exam, **do not attend** the exam.
  - I will not accept medical special consideration claims from people who have already attempted the exam.
- **Final Mark by Harmonic Mean:**
  - Final mark = $\frac{\boldsymbol{2*(ass1+ass2+ass3+proj1+proj\ 2)*finalexam}}{\boldsymbol{ass1+ass2+ass3+proj1+proj2+finalexam}}$

Course Information(cont)

- **Text Book:**
  - Elmasri & Navathe, *Fundamentals of Database Systems*, Benjamin/Cummings, 6th Edition, 2010.
- **Reference Books:**
  - J. D. Ullman & J. Widom, *A First Course in Database Systems*, Prentice Hall, 1997.
  - R. Ramakrishan, *Database Management Systems*, McGRAW-HILL, 1997.
  - D. Maier, *The Theory of Relational Databases*, Computer Science Press, 1983.

Course Outline

Time	Contents
Week 1	Subject Introduction, Conceptual DB Design (ER)
Week 2	1) Relational Data Model, 2) ER to Relational Data Model, 3) Relational Algebra
Week 3	SQL I
Week 4	PLpgSQL, Functional Dependencies
Week 5	Normal Forms, Relational DB design I
Week 6	Relational DB design II
Week 7	UML; Disks, Files,
Week 8	Transaction Management
Week 9	Graph Data Processing: Overview
Week 10	Graph Data Processing: Cohesive subgraphs
Week 11	Cloud Computing Environment
Week 12	Revision

Introduction

- Database Applications:
  - Banking System,
  - Stock Market,
  - Transportation,
  - Social Network,
  - Marine Data Analysis,
  - Criminal Analysis and Control,
  - Now, BIG DATA....

Introduction

Intelligent Transportation

Business Services

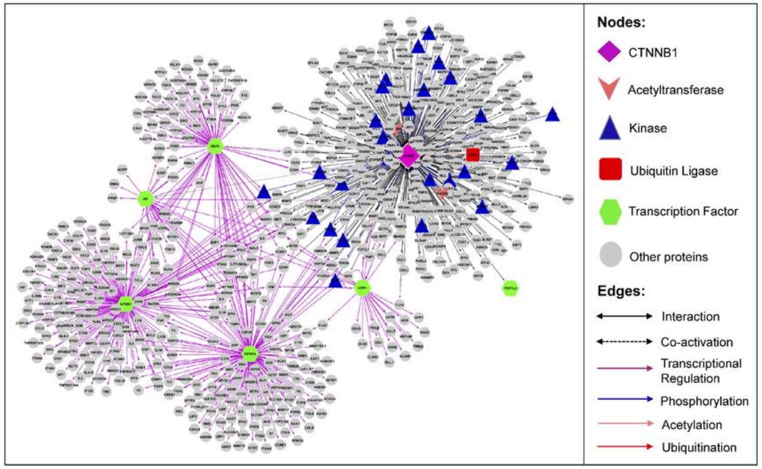
Natural Disasters

Public Health

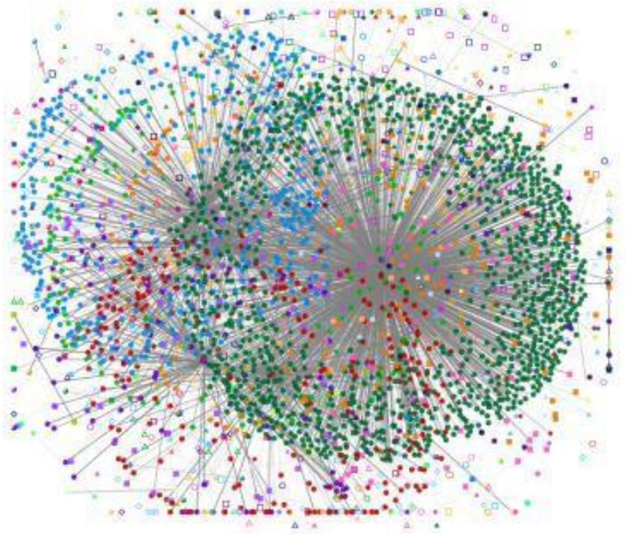
Modern Military

Tourism Development

Every where

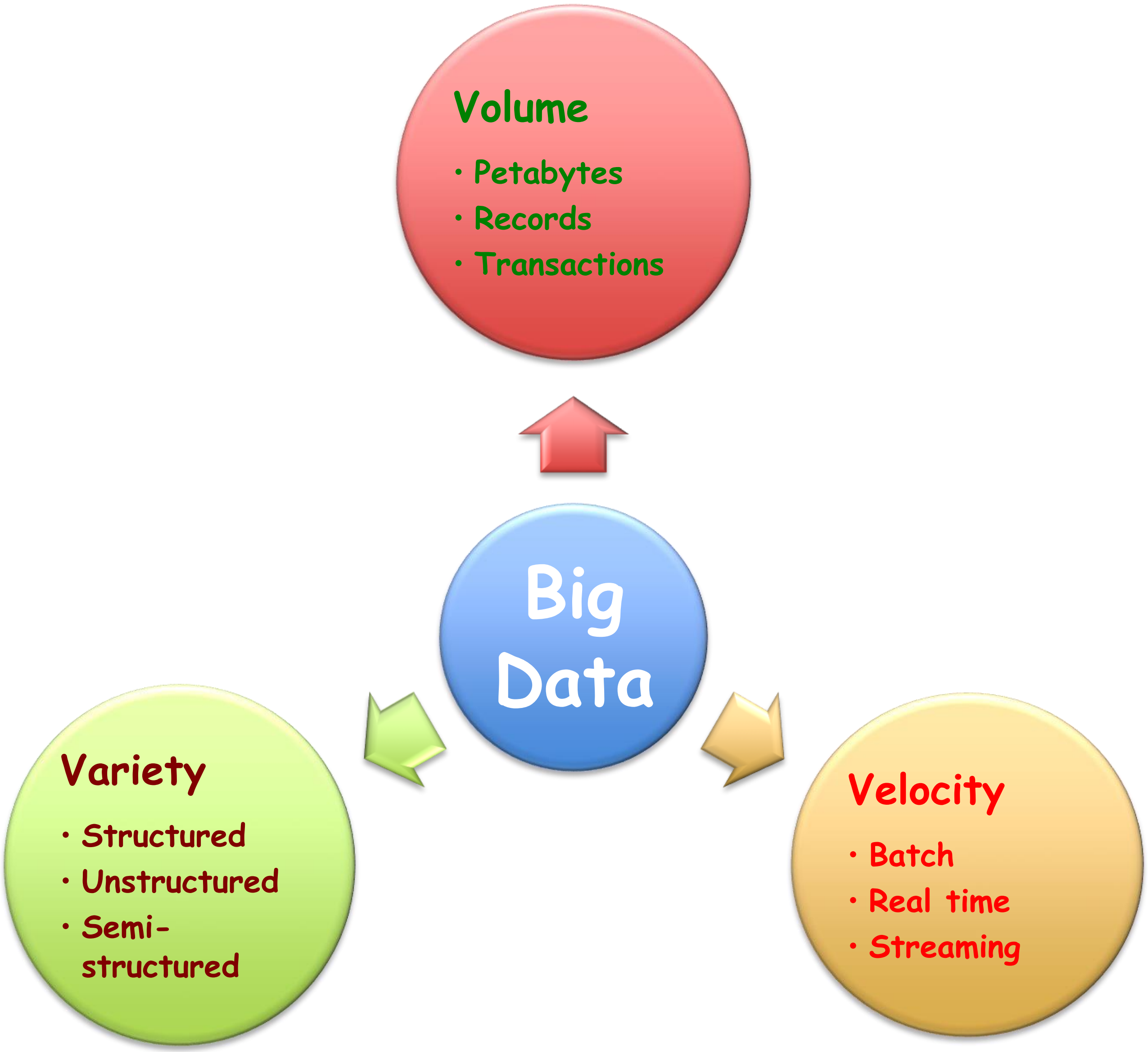


Beta-Catenin Biological Network



Web Graph

Social Network



Big Data

Volume

- Petabytes
- Records
- Transactions

Variety

- Structured
- Unstructured
- Semi-structured

Velocity

- Batch
- Real time
- Streaming

Major Research Issues

New Computing Platform/Architecture

New Graph Analytics Models

New Processing Algorithms & Indexing Techniques

Graph Processing System

- Primitive operators
- Query language
- Distributed techniques
- Storage
- etc

Introduction(cont)

- Develop a *good* database system:
  - Effectively organize data (database design).
  - Efficiently execute users queries (transaction management).
- These are even more important in modern applications, e.g. internet:
  - Huge unstructured information is available in the internet.
  - Must access the information efficiently and effectively

What is data?

- *Data* - (Elmasri/Navathe):
  - known facts that can be recorded and have explicit meaning . . .
- *Example* - a student records database:
- Contents - Information identifying students, courses they are

Item	Type of data	Stored as
Family name	String	Character strings?
Birthdate	Date	3 integers?
Weight	Real number	Floating point number?
…		

What is a database?

- Elmasri/Navathe:
  - . . . a collection of related data . . .
- Data items alone are relatively useless.
- We need the data to have some structure.
- Database can be manipulated by a database management system.

What is a database management system(DBMS)?

- Elmasri/Navathe:
  - *DBMS*: . . . a collection of programs that enables users to create and maintain a database . . .
  - *Database system*: . . . The database and DBMS together . . .

Database requirements

- Database system provides facilities to:
  - *Define a database* - specifying the data items to be stored and their types,
  - *Construct a database* - loading the data items and storing them on some storage medium (usually disk),
  - *Manipulate a database*
    - querying - i.e. retrieving relevant data,
    - updating - i.e. adding, deleting or modifying data items:
      - from one “correct” state to another “correct” state,
  - *reporting*

Database requirements(cont)

- Database system must be
  - *Timely* - e.g. an airline database (fast response), a CAD system (must be interactive),
  - *Multi-user* - e.g. trading system,
  - *Modifiable* - must be able to be extended or reorganised, e.g. to cope with new laws, requirements, business conditions,
  - *Secure* - different classes of users may need different levels of access,
  - *No redundancy*,
  - *Robust* - e.g. power failure during an update - must be able to recover to a consistent state.

Database requirements(cont)

- A database system must address these issues and provide solutions - DBMS:
  - *a special purpose DBMS,*
  - *a general DBMS.*
- **The DBMS solution vs meta-data**
- To allow a general DBMS to be applied to a particular database application, we need

meta-data

Database requirements(cont)

- *Meta-data*: a definition and description of the stored database, such as structure of each file, type and storage format of each data item, constraints etc.
- Stored in the system *catalog*.

Benefits of meta-data

- *program-data independence* - DBMS access programs may be written independent of file structures and storage formats,
- *data abstraction* - information hiding.
  - Users are provided with a *conceptual representation* of the data using a high level *data model*.
- *support for views* - different users can have different views of the database. e.g.
  - salary details may be hidden from some users,
  - statistical summaries may be derived and appear as stored data for some users.

Database personnel

- *Database Administrator(DBA)* - This person is responsible for the centralised control of the database:
  - authorising access
  - monitoring usage,
  - recovery,
  - identifying the data,
  - choosing appropriate structures to represent and store the data,
  - managing definitions of views . . .

Database personnel(cont)

- *End user* - People requiring access to the database for querying, updating, reporting etc.
  - Naive (parametric) user - typically use the database via “canned transactions” - standardised queries and updates, often through a menu system of some kind,
  - Online user - has an understanding of the database system. May be capable of designing their own queries etc.

Database personnel(cont)

- *Systems analyst*:
  - determine end users requirements,
  - develop specifications for canned transactions and reports,
  - may also take part in database design.
- *Application programmer* - Implements the specifications given by analyst:
  - tests,
  - debugs,
  - maintains the resulting programs.

DBMS concepts

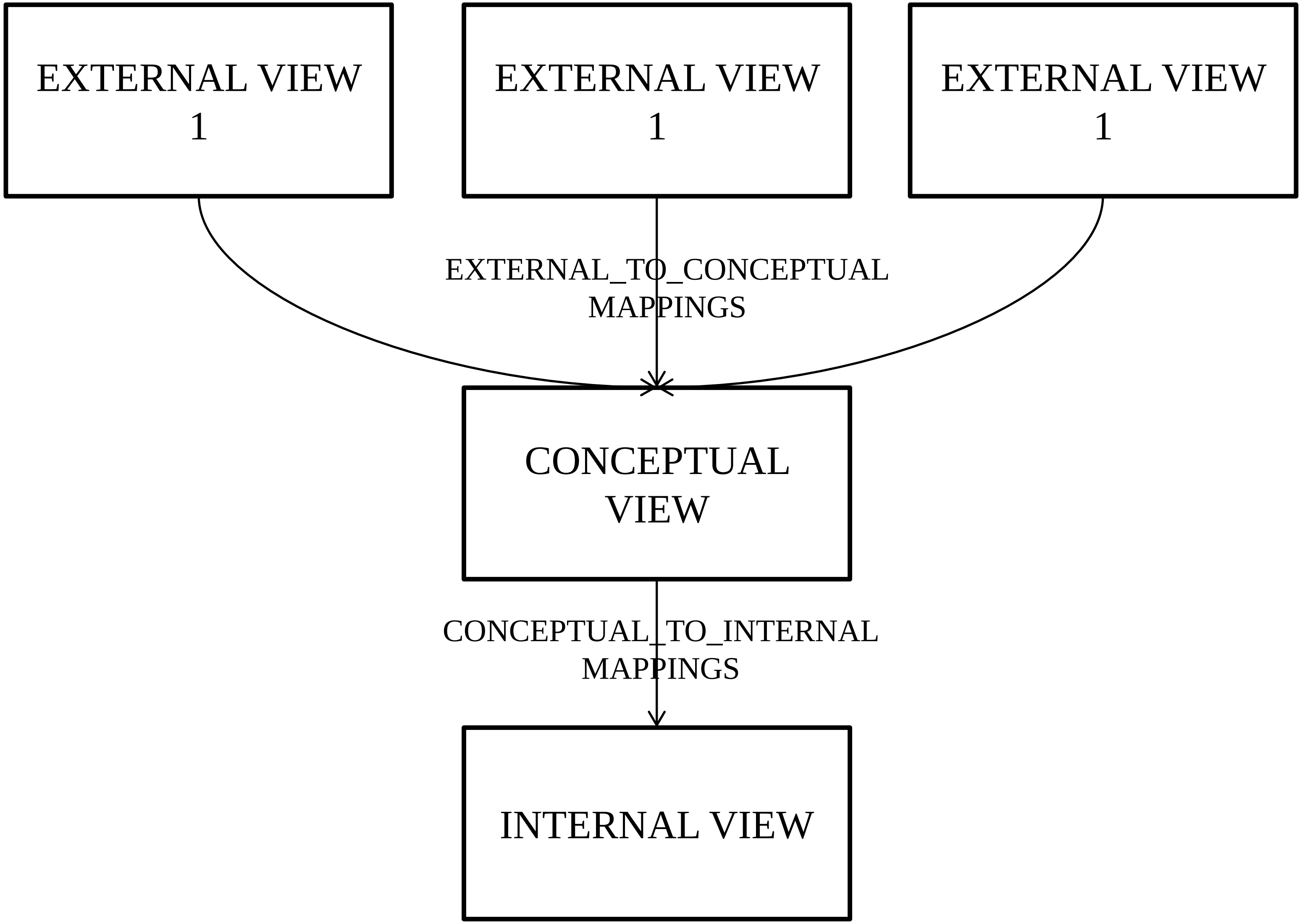
- *Data model*: a set of concepts that is used to describe the allowed structure of a database. i.e. the structure of the meta-data.
- May be classified as:
  - High-level or conceptual (e.g. ER model – concerns entities, attributes and relationships)
  - Implementation or record-based (e.g. Relational, Network, Hierarchical - suggests a physical implementation)
  - Low-level or physical (concerns record formats, access paths etc)

DBMS concepts(cont)

- *Database Schema:* An instance of a data model, that is, a description of the structure of a particular database in the formalism of the data model. (Intention)
- *Database Instance (or State):* The data in the database at a particular time. (Extension)
- In these terms:
  - We define a database by specifying its schema.
  - The state is then an empty instance of the schema.
  - To create the initial instance we load in data.
  - After this, each change in state is an update.

ANSI-SPARC three level architecture

- ANSI: American National Standard Institute.
- SPARC: Standards Planning and Requirements Committee.
- ANSI-SPARC three level architecture (1975-1977):
  - The *external* or *view level* includes a number of external schemas or user views.
  - The *conceptual level* has a conceptual schema, which describes the structure of the whole database for a community of users.
  - The *internal level* has an internal schema, which describes the physical storage structure of the database.



EXTERNAL VIEW 1

EXTERNAL VIEW 1

EXTERNAL VIEW 1

EXTERNAL_TO_CONCEPTUAL MAPPINGS

CONCEPTUAL VIEW

CONCEPTUAL_TO_INTERNAL MAPPINGS

INTERNAL VIEW

ANSI-SPARC three level architecture(cont)

- 3 levels of abstraction => 2 levels of data independence:
  - *logical data independence*: the ability to change the conceptual schema without changing external views. Must change the external-to-conceptual mapping though.
  - *physical data independence*: the ability to change physical storage paths and access structures without changing the conceptual view. Must change the conceptual-to-internal mapping though.

Database languages

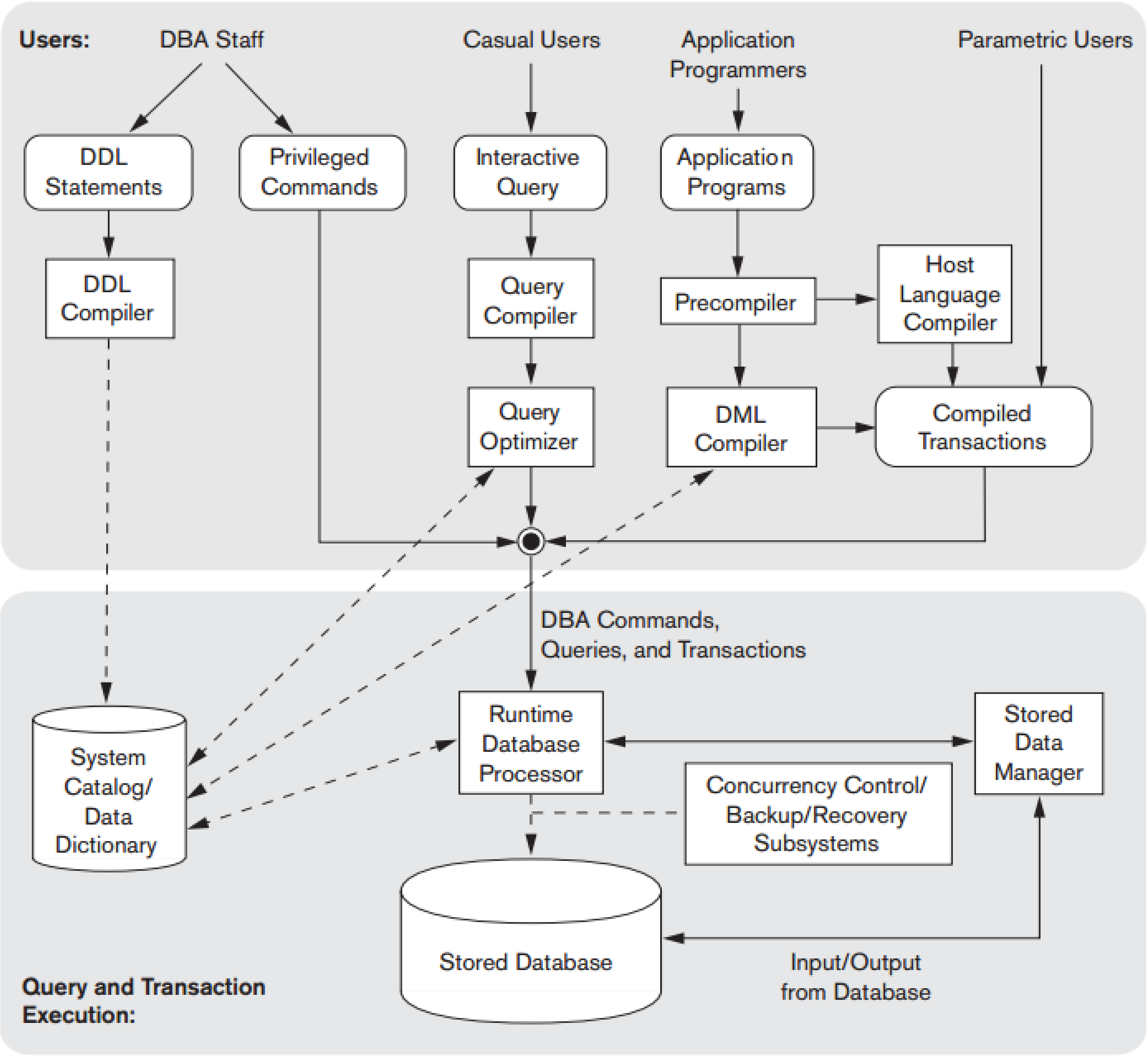
- In the three level architecture:
  - *Data definition language (DDL):* used to define the conceptual schema.
  - *View definition language (VDL):* used to define external schemas.
  - *Storage definition language (SDL):* used to define the internal schemas.
- In DBMS where conceptual and internal levels are mixed up, DDL is used to define both schemas.

Database languages(cont)

- *Data manipulation language (DML):* used to construct retrieval requests (queries) and update requests:
  - Low-level or procedural
    - embedded in a general purpose language,
    - record at a time
  - High-level or non-procedural
    - interactive and/or embedded
    - set at a time/ set oriented.
- In most current DBMSs, a comprehensive integrated language is used; for example SQL.

Database components

- See Fig2.3 in Elmasri/Navathe.
- *Run-time database processor* - Receives retrieval and update requests and carries them out with the help of the stored data manager.
- *Stored data manager or file manager* - Controls access to the DBMS information stored on disk:
  - may use the OS for disk access,
  - controls other aspects of data transfer, such as handling buffers.
- *Pre-compiler* - Extracts DML commands from the host language program.
  - These are compiled by the DML compiler, the rest is compiled by the host language compiler, then they are linked to produce executable code with calls to the data manager.
- *Query processor (or Complier)* - Parses high-level queries and converts them into calls to be executed by the data manager.



Component modules of a DBMS and their interactions.