



# COMP9311 DATABASE SYSTEMS

---

- Wenjie Zhang
  - School of Computer Science and Engineering
  - Office: K17-502
  - E-mail: [wenjie.zhang@unsw.edu.au](mailto:wenjie.zhang@unsw.edu.au)
  - Tel: (+61 2) 93857799
  - <http://www.cse.unsw.edu.au/~zhangw>
- 

- **www home address of 9311:**
- <http://www.cse.unsw.edu.au/~cs9311>

# Course Information

---

## Lectures:

18:00 - 20:00 (Mon)

week 1 – 3, 5 – 10

18:00 - 20:00 (Thu)

week 1 – 9

Location: Keith Burrows Theatre (K-J14-G5)

9 weeks lectures in total.

**Lab:** week 2 – 10

**Consultation:** 15:00 – 16:00 (Mon)

➤ Location : K17-401K week 2 – 3, 5 – 10

**Q&A Forum:** <https://groups.google.com/group/comp9311-19t3>

**Course Email:** comp9311unsw@gmail.com

For normal questions, we recommend you use the Q&A forums. You are also welcome to contact us via course email if something is urgent.

# Course Information

---

- Lecturer-in-Charge (LiC)
  - A/Prof. Wenjie Zhang
  - Office: K17 502, Phone: 93857799
  - Email: [wenjie.zhang@unsw.edu.au](mailto:wenjie.zhang@unsw.edu.au)
- Tutors
  - A group of PhD students in the *Data and Knowledge Research Group*
  - COMP9311 in T3 has ~420 students, each tutor looks after 1-2 labs.
- Course Website
  - <http://www.cse.unsw.edu.au/~cs9311>

# Why study databases

---

Every significant modern computer application has Large Data.

This needs to be:

- **stored** (typically on a disk device)
- **manipulated** (efficiently, usefully)
- **shared** (by many users, concurrently)
- **transmitted** (all around the Internet)

**Red** stuff handled by databases; **brown** by networks.

Challenges in building effective databases: efficiency, security, scalability, maintainability, availability, integration, new media types (e.g., music, video), ...

# Syllabus Overview

---

Data modelling and database design (Week 1 to Week 2)

- ER model, ODL, ER-to-relational
- Relational model (relational algebra), mapping of ER to relational model

Database application development (Week 3 to Week 4)

- SQL, views, stored procedures, triggers, aggregates
- PostgreSQL: PLpgSQL (procedural)

Formal database design theory and database system architecture (Week 5 – Week 8)

- Normalisation, functional dependencies
- Storage and indexing, data access operations
- Query processing: translation, optimisation, evaluation
- Transaction processing: transactions, concurrency control, recovery

Research Topics of Databases (Week 9)

- Cutting-edge technologies for database management

Revision (Week 10)

# Course Outline

---

| Time    | Monday  | Thursday                          |
|---------|---|-----------------------------------|
| Week 1  | Subject Introduction, Conceptual DB Design (ER) | Relational Data Model             |
| Week 2  | ER to Relational Model Mapping                  | Relational Algebra                |
| Week 3  | SQL   | SQL (continue)                    |
| Week 4  | Public Holiday No Lecture                       | PLpgSQL                           |
| Week 5  | Functional Dependencies                         | Normal Forms                      |
| Week 6  | Relational DB Design                            | Disks, Files, Index               |
| Week 7  | Transaction Management                          | Transaction Management (continue) |
| Week 8  | Query Processing                                | NoSQL Data Models                 |
| Week 9  | Research Topics                                 | Research Topics (continue)        |
| Week 10 | Revision  | No lecture                        |

# Textbook

---

Lecture notes are sufficient but the following text books are good:

## **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. Ramakrishnan, *Database Management Systems*, McGRAW-HILL, 1997.
- D. Maier, *The Theory of Relational Databases*, Computer Science Press, 1983.

# Teaching/Learning Approaches

---

## What we do to support your learning:

- Face-to-face support sessions in labs
  - *Lab exercises (starting Week 2, every week)*: guides you through the practical skills on the database application programming part of the course
  - **We run consultations in the lab: tutors will answer all sorts of questions related to the course.**
  - Weekly course admin consultations (every Monday before the class)
- Extra exercise questions with answers: available in the course website
- Online support
  - *course forums* (<https://groups.google.com/group/comp9311-19t3> link available in the course website)
  - send an email to [comp9311unsw@gmail.com](mailto:comp9311unsw@gmail.com) (regularly monitored)



# Teaching/Learning Approaches

---

Things that we expect you to **do**:

- Follow the lecture content
- *Practice the exercise questions*: exercise questions with solutions covering the key topics. Try to solve them first and before checking the answers
- *Practical work*: lab exercises. Attending labs is strongly encouraged.

*You will show us your progress/learning outcomes through:*

- *2 Assignments*: extended exercises
- *1 Project*: practical questions in SQL and PLpgSQL
- *Final exam*: paper-based

# Assignment and Project

---

2 assignments, 1 project. All individual work !

Assignments (50%):

- Ass 1: Data Modelling + Relational Algebra (25%) (week 2-4)
- Ass 2: DB Design Theory + Database Storage Structures + Transaction (25%)  
(week 8-10)

Projects (50%)

- Proj 1: SQL PLpgSQL (50%) (week 4-7)

**Penalty for later submissions:** *20% reduction per day.*

# Exam and Marks

---

**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 =  $\frac{2*(ass1+ass2+proj1)*final\ exam}{ass1+ass2+proj1+final\ exam}$

# Database related courses in CSE

---

COMP9311 introduces foundations & technology of databases

- skills: how to build database-backed applications
- theory: how do you know that what you built was good

After COMP9311 you can go on to study ...

- COMP9315: how to build relational DBMSs (write your own PostgreSQL or Oracle)
- COMP9318: techniques for data mining (discovering patterns in DB)
- COMP9319: Web data compression and search (dealing with a large amount of Web data)
- COMP6714: information retrieval, web search (dealing with text data)
- COMP932[1|2|3]: service-oriented computing, which relies on DB background

# Beyond the courses ...

---

**Research Project:** 12 UoC or 18 UoC to focus on a specific research project

- COMP9945: <https://www.handbook.unsw.edu.au/postgraduate/courses/2019/COMP9945>
- COMP9596: <https://www.handbook.unsw.edu.au/postgraduate/courses/2019/COMP9596>

**Research Degrees:** PhD (3 – 3.5 years), Master by Research or Master of Philosophy (2 – 2.5 years) . <https://research.unsw.edu.au/higher-degree-research-programs>

- To deeply explore on research problems
- Feel free to contact me if you are interested in obtaining a research degree

# 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



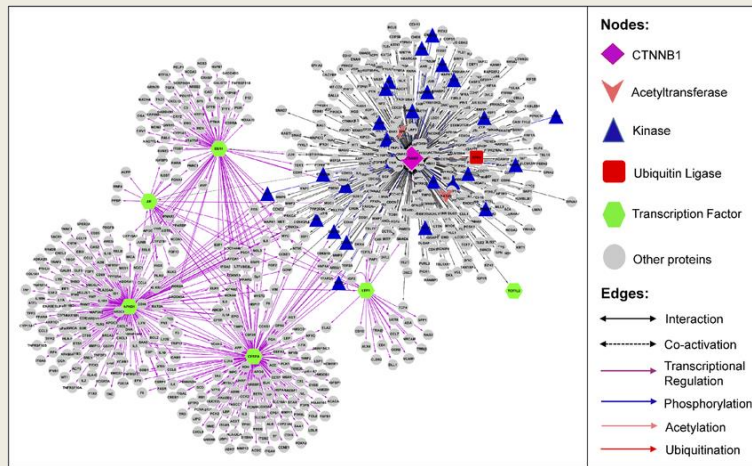
## Modern Military



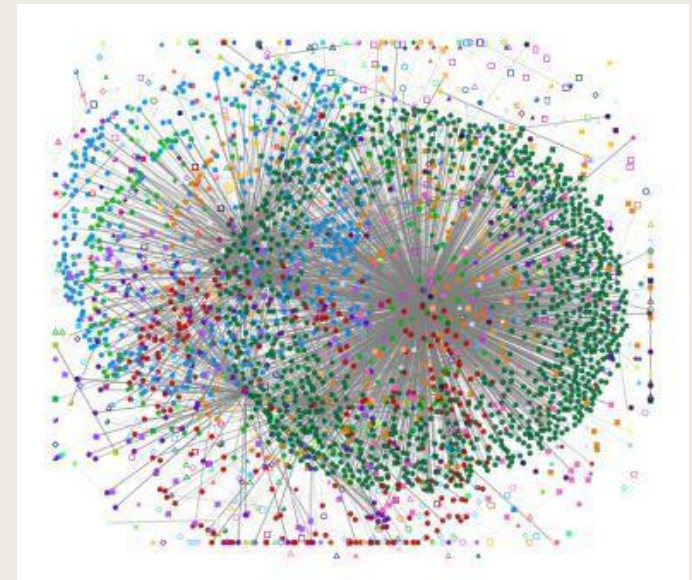
## Tourism Development



# Introduction



Beta-Catenin Biological Network



Web Graph



Social Network



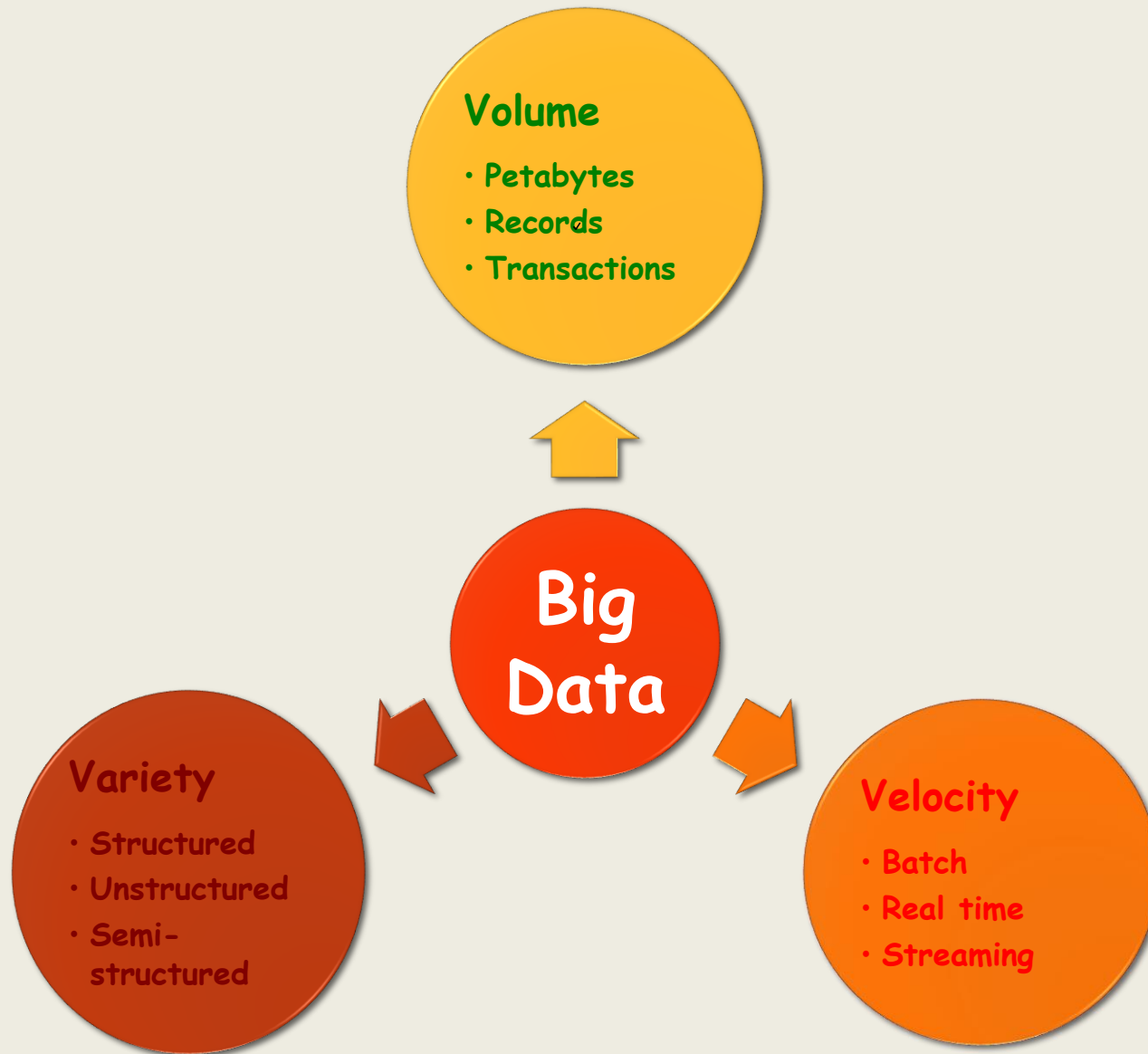


# Databases: Important Themes

---

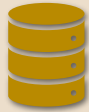
The field of *databases* deals with:

- *data* ... representing application scenarios
- *relationships* ... amongst data items
- *constraints* ... on data and relationships
- 限制 ◦ *redundancy* ... one source for each data item
- *data manipulation* ... declarative, procedural
- *transactions* ... multiple actions, atomic effect
- *concurrency* ... multiple users sharing data
- *scale* ... massive amounts of data



# 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 enrolled in, results from past courses . . .

| 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*

查询 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<sub>(cont)</sub>

---

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*, 冗长
- *Robust* - e.g. power failure during an update - must be able to recover to a consistent state.



# Database requirements<sub>(cont)</sub>

---

A database system must address these issues and provide solutions - DBMS:

- *a special purpose DBMS,*
- *a general DBMS.*

# 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<sub>(cont)</sub>

---

*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<sub>(cont)</sub>

---

## *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 概念的 conceptual (e.g. ER model – concerns 实体 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<sub>(cont)</sub>

---

纲要

*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

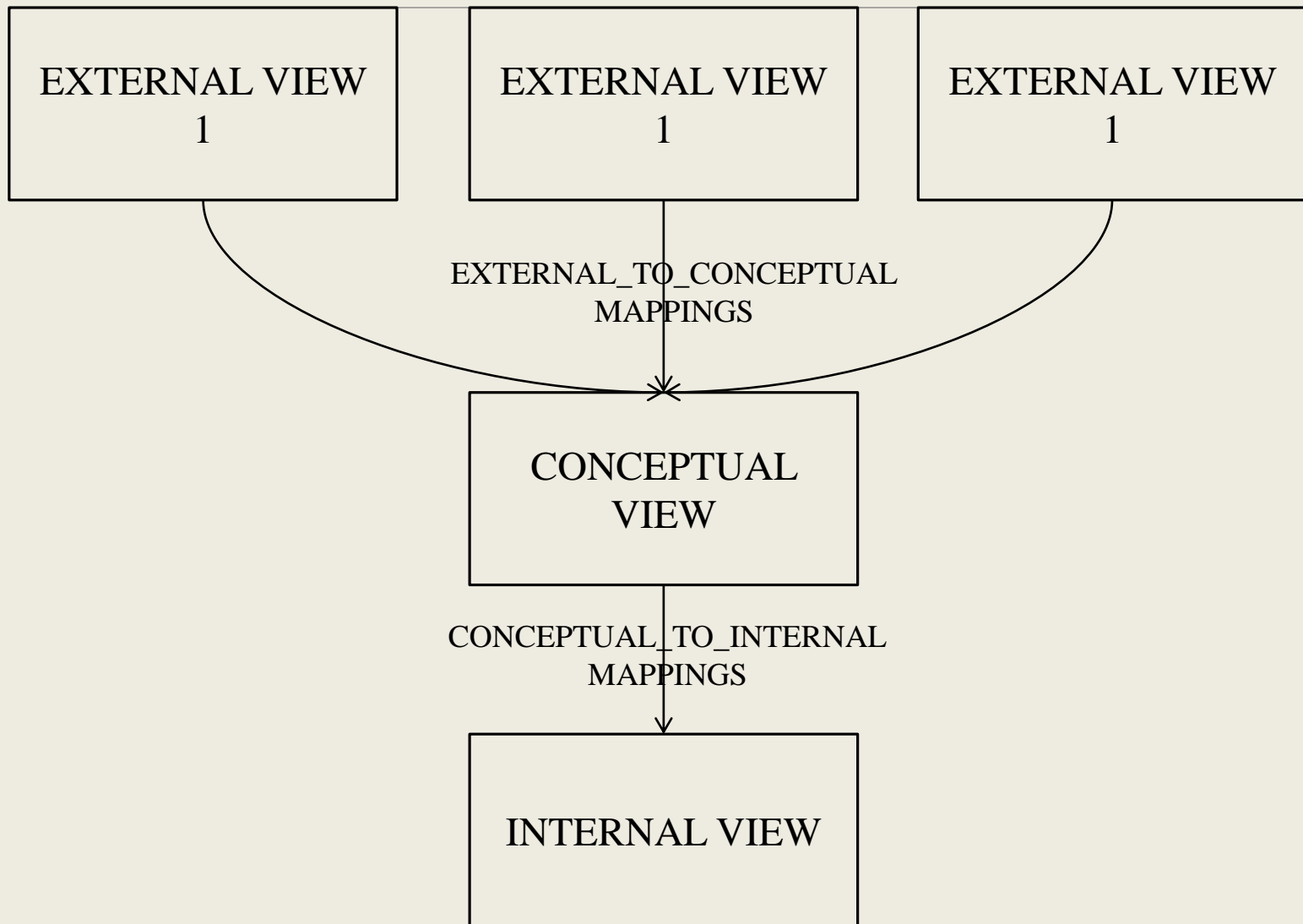
---

ANSI: American National Standard Institute.

SPARC: Standards Planning and Requirements Committee.

ANSI-SPARC three level architecture (1975-1977):

- The 外在的 *external* or *view level* includes a number of external schemas or user views.
- The 概念的 *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.





# ANSI-SPARC three level architecture<sub>(cont)</sub>

---

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<sub>(cont)</sub>

---

*Data manipulation language (DML)*: <sup>操作</sup>used to construct <sup>检索</sup>retrieval requests (queries) and update requests:

- Low-level or procedural
  - <sup>嵌入</sup>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

---

*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:

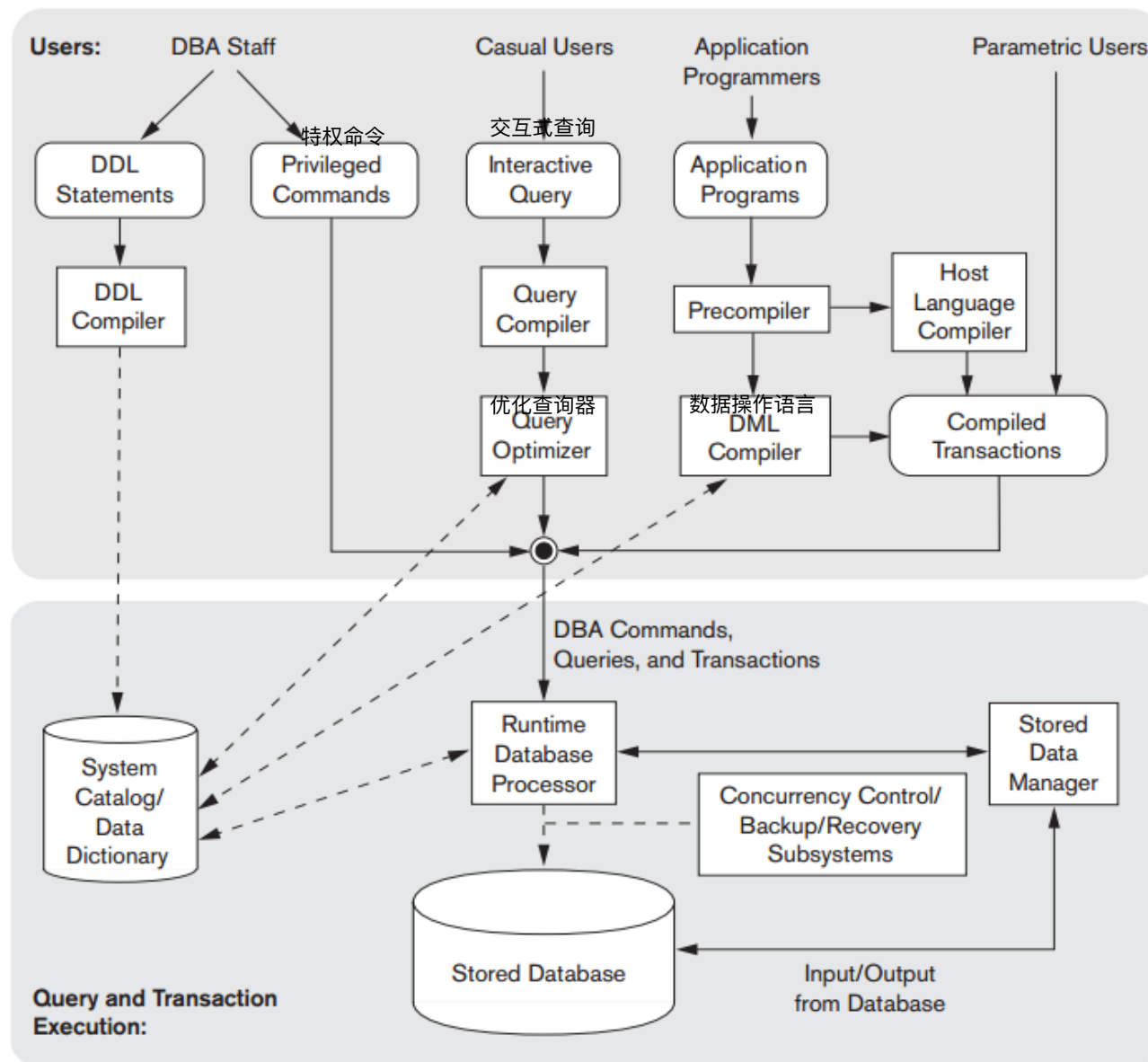
- may use the OS for disk access,
- controls other aspects of data transfer, such as handling buffers.

获得

*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 Compiler)* - Parses high-level queries and converts them into calls to be executed by the data manager.



Component modules of a DBMS and their interactions.

# Learning Outcomes:

---

- Introduction of the course
- General understanding of terms in databases, details will be found in later part of the course.