# Introduction to Database Systems (2)

SWEN 304

Trimester 2, 2019

Lecturer: Dr Hui Ma

**Engineering and Computer Science**

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

slides by: Pavle Mogin & Hui Ma

# Outline

- Data models
- Schemas and Instances
- The Three-Schema Architecture
- Program - Data Independence
- Data manipulation languages
  - Navigational
  - Declarative
- Reading: Chapter 2 of the textbook

# Data Models

- A 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

- A data model is 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 this real system

# Data Models

- Constructs are used to define the database structure
- Constraints are statements about values and relationships that must hold between data
- Operations: 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. update_inventory, compute_student_gpa

# Classification of Data Models

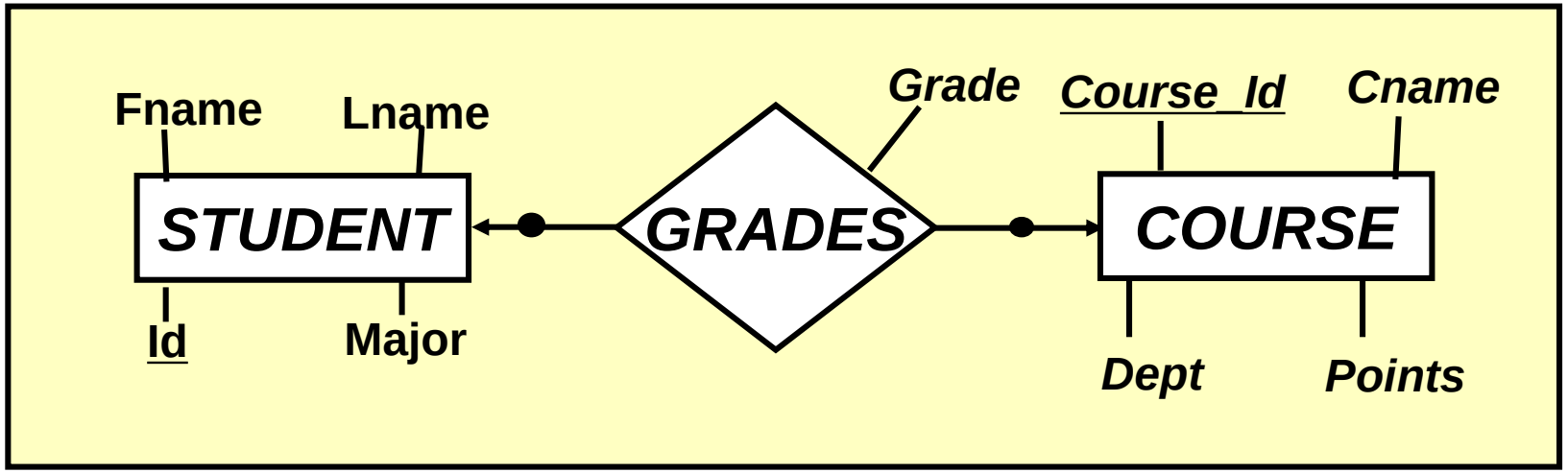
- Conceptual (high level, semantic) data models:
  - Provide concepts that are close to the way many users perceive data
  - Entity based or object-based data models
- Physical (low-level, internal) data models:
  - Provide concepts that describe details of how data is stored in the computer
- Implementation (Representational) data models
  - Provide concepts that fall between the above two, balancing user views with some computer storage details
  - Relational data model: used in several commercial products (DB2, ORACLE, SQL Server, SYBASE, INFORMIX)
  - Legacy data models: network model, hierarchical model

# Schemas and Instances

- A database schema (also intension): the abstract description of a database
  - includes the descriptions of the **database structure** and the **constraints** that should hold on the database.
  - is a fairly static
  - given in the context of a particular data model and a corresponding data definition language
- A schema diagram: an illustrative display of (most aspects of) a database schema
- A schema construct: a **component** of the schema or an object within the schema,
  - e.g., STUDENT, COURSE

# Example Database Schemas

## EER Schema Diagram




## RELATIONAL Schema

STUDENT (Id, Lname, Fname, Major),
GRADES (Id, Course_id, Grade),
COURSE (Cname, Course_id, Points, Dept)

# Schemas and Instances (continued)

- A database instance (also extension or state of a database): the actual data stored in a database at a *particular moment in time*
  - is produced by populating (loading) schema description by data
  - corresponds to schema structure, and
  - satisfies all schema constraints
- A database instance should reflect changes of the real system's state, and that is accomplished by its updating
- a schema instance is a dynamic category

# A Database Instance

| STUDENT | | | |
|---|---|---|---|
| **Id** | **Lname** | **Fname** | **Major** |
| 300111 | Smith | Susan | COMP |
| 300121 | Bond | James | MATH |
| 300132 | Smith | Susan | COMP |

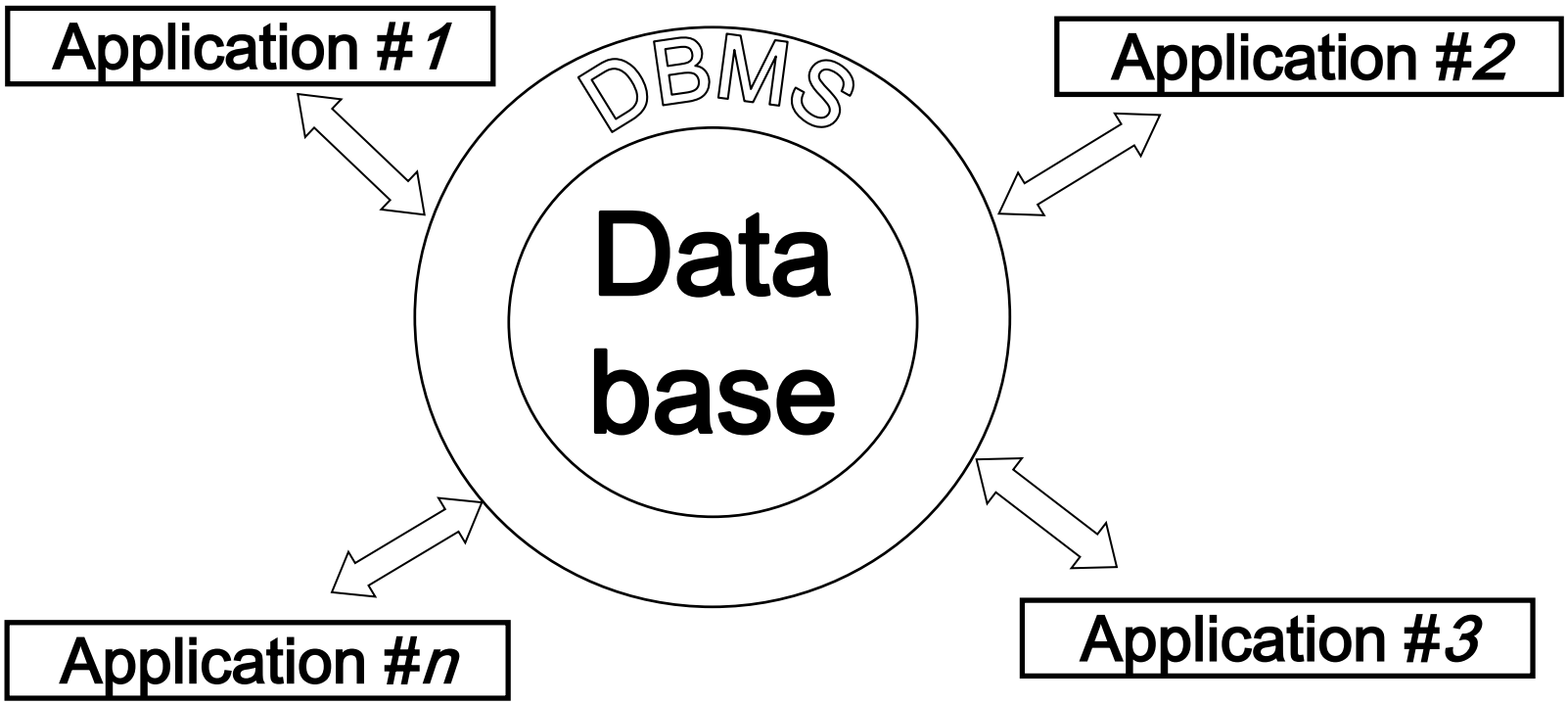
| COURSE | | | |
|---|---|---|---|
| **Course_id** | **Cname** | **Points** | **Dept** |
| COMP302 | DB sys | 15 | Engineering |
| COMP301 | softEng | 20 | Engineering |
| MATH214 | DisMat | 15 | Mathematics |

| GRADES | | |
|---|---|---|
| **Id** | **Course_id** | **Grade** |
| 300111 | COMP302 | A+ |
| 300111 | COMP301 | A |
| 300111 | MATH214 | A |
| 300121 | COMP301 | B |
| 300132 | COMP301 | C |
| 300121 | COMP302 | B+ |
| 300132 | COMP302 | C+ |

# Data Organization Approaches

- Suppose UoD functions (like HR, Accounting, Sales, Payroll,…) are supported by an application, containing a number of programs
- There are two characteristic approaches to data organization and storage in the realm of an information system:
  - Traditional approach: each application contains all necessary data, as its own, organized into files
  - Database approach: all applications use a common database (no application owns the database)

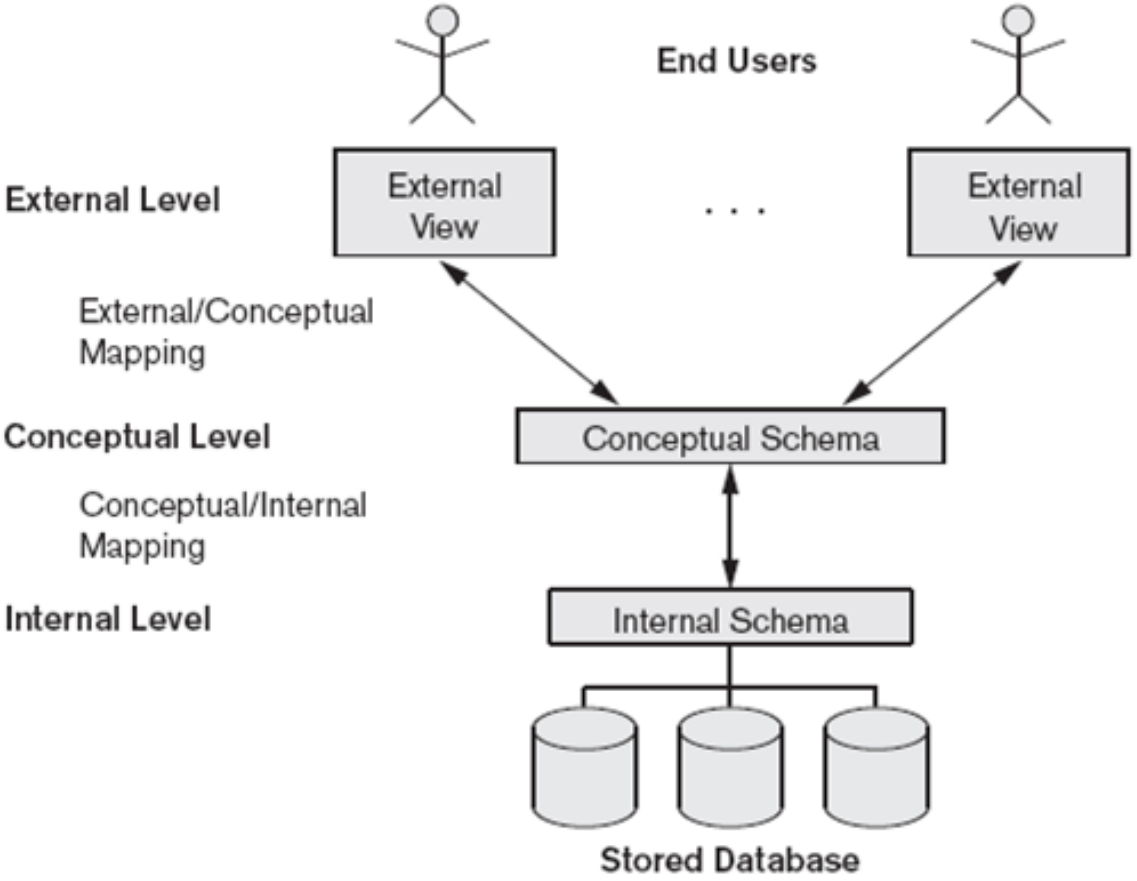
# Database Approach



# Data Independence

- In the traditional (file) approach to data organization, information about a file physical structure was embedded into programs
- The same was true for the first (prerelational) database systems
- Changes of a file structure induced changes in programs
- This phenomenon is called (program -) data dependence
- To overcome the problem, we introduce procedures for **physical and logical data independence**

# Three-Schema Architecture




**(Elmasri & Navathe, 2011)**

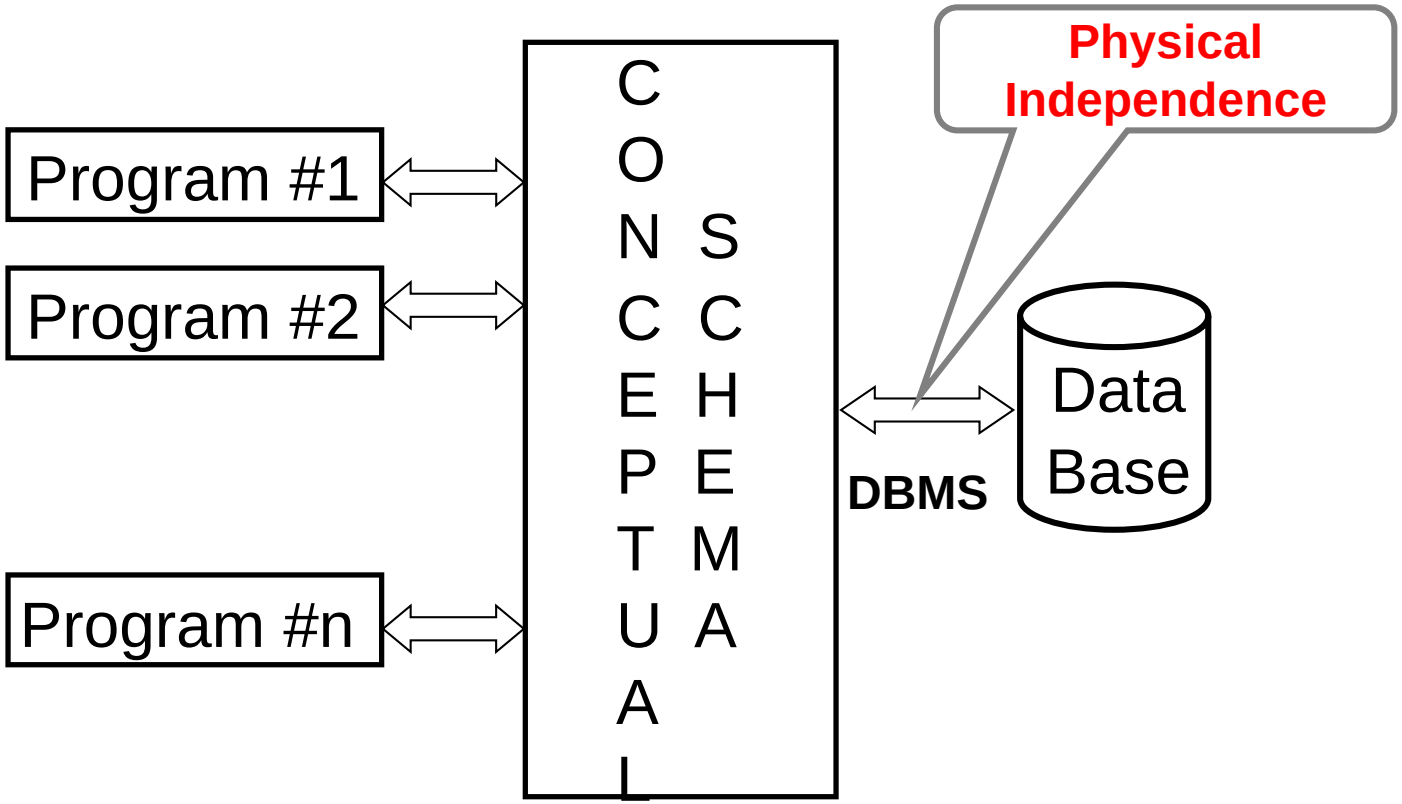
# Three-Schema Architecture

- External Schemas:
  - perspective of a user or an application program accessing the database
  - describes restructured parts of the database (views) that are used in a particular application
  - mapped to logical schema
- Logical or Conceptual Schemas:
  - perspective of the DBMS
  - abstract, integrated description of all data independent from their implementation
  - mapped to physical schema
- Physical or Internal Schemas:
  - perspective of the implementation/system realization
  - describes physical storage and access structures

# Physical Data Independence

- 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

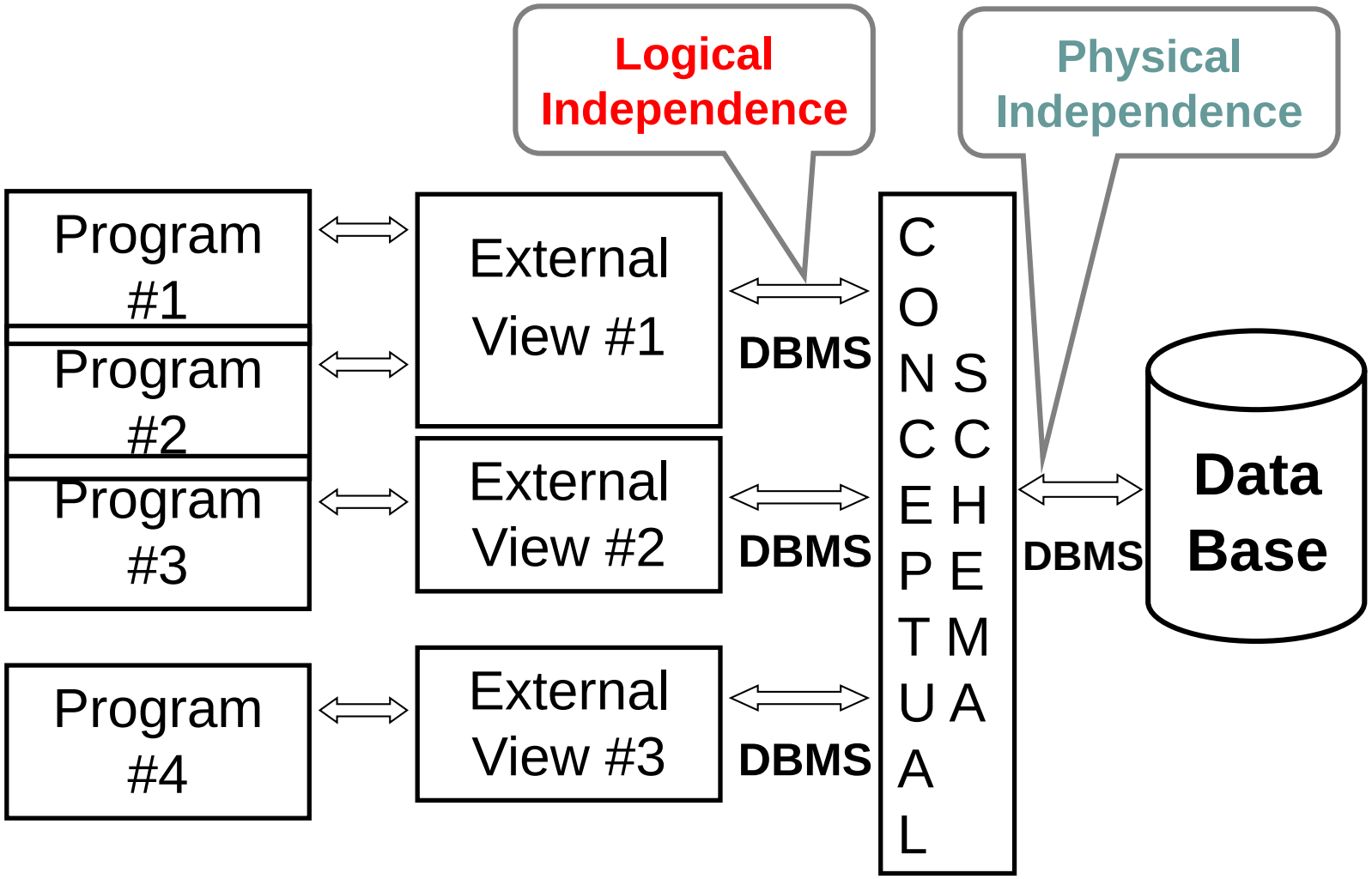
# Physical Data Independence



# Logical Data Independence

- 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 (unfortunately computationally hard to achieve: view update problem)

# Logical Data Independence



# Languages used in Data Management

- It is common to distinguish language according to their specific purpose:
  - A data definition language (DDL) is used to define the database schemata
  - A data manipulation language (DML) is used to update the data in the database (insert, delete, modify)
  - A query language is used to access the data in the database and to retrieve data

# Data Manipulation Language

- Used to retrieve, insert, delete, and modify data
- 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

- A 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

- A 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

- Query: "Retrieve all Courses and Grades of the student with Id = 300111"

- Simplified navigational pseudo code:

```
Find record with Id= 300111 in GRADES
If successful then
    Do while there are courses connected to the student
        Find next course Id in GRADES
        Find corresponding Grade
        Find Course name in COURSE
    End do
Else
    Display error message
End if
```

# Navigational Versus Declarative DML

- Query: "Retrieve all Courses and grades of the student with Id = 300111"

- A fully declarative program:

```
SELECT Course_id, Cname, Grade
FROM COURSE C, GRADES G
WHERE Id = 300111 AND G.Course_id = C.Course_id;
```

# Database Users and Languages

- Database administrator (DBA):
  - DDL describes 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

# Summary

- A **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
- A **database instance** refers to the actual data stored in a database at a *particular moment in time*
- The **three schema architecture** (external, conceptual, and internal schemas) – introduced to enable logical and physical data independence
- Main advantages of the database over traditional approach:
  - Program – data independence,
  - Data consistency,
  - DBMS controlled data sharing and recovery

# Plan for the Next Lecture

- Introduction to the relational data model – motivations and basic ideas
- Basic terms and concepts of the relational data model
- Relational schemas and instances
- Constraints of the relational data model
- Reading:
  - Sections 5.1 and 5.2, Chapter 5 of the textbook

# Set Notations

- We need set notation to represent formal definitions in this course
- A set is several things considered together as one thing
- There are two ways to specify a set
  - $\{x_1, \ldots, x_n\}$, list all the elements in a set
    - {}, the empty set
    - {Wellington, Auckland, Christchurch}
    - {{*A*}, {*A*, *B*}, {*A*, *B*, *C*}}, a set of sets
  - $\{x|x \in \varphi\}$, describe the members that satisfy a property
    - $\{x|x \in \mathbb{N}\}$, the set of natural numbers
    - {students | all the students enrolled in SWEN304}

# Set Notations

- If two sets have the same members, they are the same set
- If one set contains something that is not in the other set, then they are different
  - e.g. {1, 2, 3} ≠ {1, 2, 4}
- The members in a set have no order
  - e.g. {1, 2} = {2, 1}
- Each element cannot be in the set more than once
  - E.g. {1, 1} = {1}

# Set Operations

- **Membership**: $x \in A$ if $x$ is in set $A$;

  $x \notin A$ if $x$ is not in set $A$
- **Equality**: $A$ and $B$ are equal if they have the same members, $A = B$
- **Subset**: if every member of $A$ is in $B$ we write $A \subseteq B$; $A$ is called proper subset of $B$ if $A \subseteq B$ and $A$ and $B$ are not equal
- **Union**: $A \cup B$ for the set containing everything in $A$ and $B$
  - e.g. {a, b c} ∪ {a, c, d, e}= {a, b, c, d, e}
- **Intersection**: $A \cap B$ for the set of elements that are in both $A$ and $B$
- **Difference**: $A - B$ for the set of elements from $A$ but not $B$

# Set Operation Exercise

- Let $A = \{x, y, z\}$, $B = \{1, 2\}$
- Which of the following are correct?
  - $\{y\} \subseteq A$
  - $z \in A$
  - $\{x, y\} \subseteq (A \cup B)$
  - $x \in (A \cap B)$
  - $x \in (A - \{y, z\})$
- $(\{x, 1\} \cup A) \cap B = ?$

# Cartesian Products of Sets

- A **n-tuple** is an ordered list of n elements
  - (Hui, 5657) is a 2-tuple, (x, y, 1, 2, 2) is a 5-tuple

- The **Cartesian product** operation takes an ordered list of sets and returns a set of tuples
- Cartesian product $D_1 \times \ldots \times D_n$ is the set of all possible combinations of values from the sets $D_1, \ldots, D_n$
- For example, $D_1$ ={Wellington, Auckland}, $D_2$ = {1, 2, 3}

$D_1 \times D_2$

={(Wellington, 1), (Wellington, 2), (Wellington, 3),
(Auckland, 1), (Auckland, 2), (Auckland, 3)}