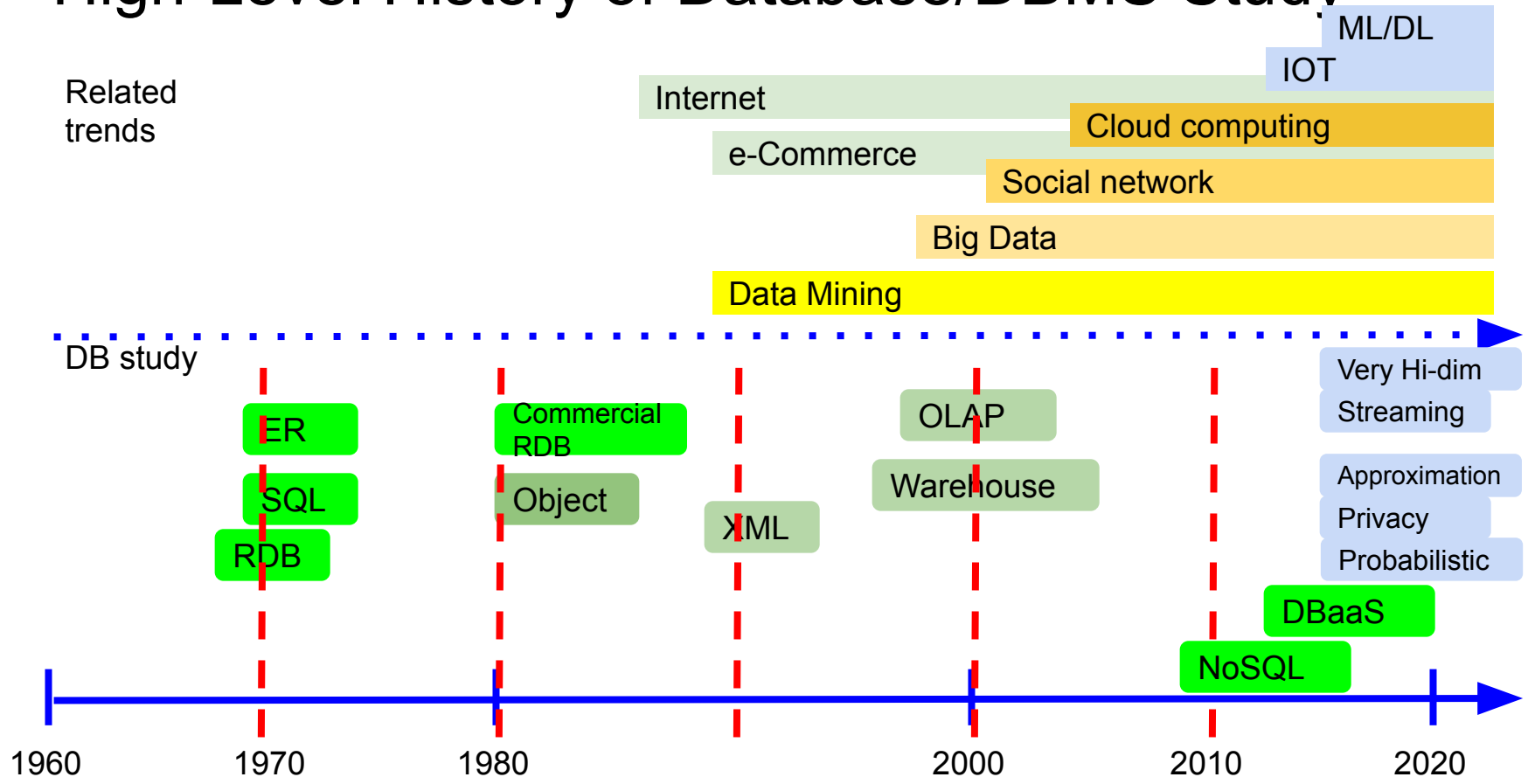
DS108 Relational Database Overview

Ming-Ling Lo
20191002

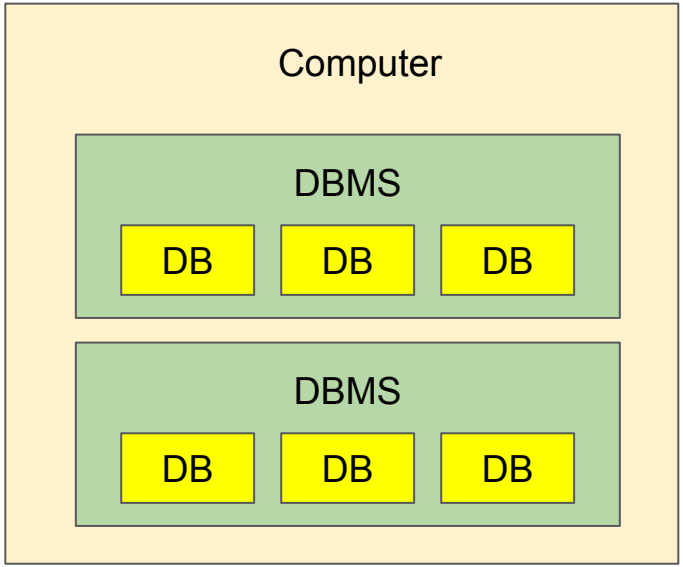
The Study of Relational Databases

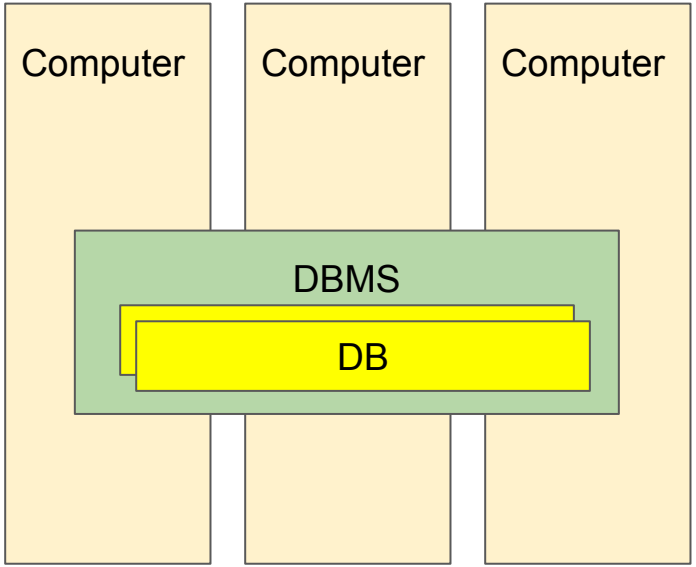
- Basics
  - Relational database model
  - SQL language
  - Relational Algebra
  - Relational Calculus
- Advanced basics
  - RDB design
    - Mapping from ER/EER to RDB, Normalization
  - Programming interfaces
    - ODBC, JDBC, Other programming
- Implementation and Performance

High-Level History of Database/DBMS Study



DB, DBMS Instances and Computers





Questions: Why storing data in the same DBMS into multiple databases?

Relational Database Model

Relational Database Model

- Used mathematical relation as its building block
- Theoretical basis in set theory and first-order predicate logic

- Intuitively
  - Think of Relation as a table
  - Row → Tuple
  - Column → Attribute
  - Data types of value → Domain
  - But **Order of rows** is considered immaterial (not counted)

ID	name
99633	John
99634	Mary
99635	Peter

ID	name
99634	Mary
99633	John
99635	Peter

Relational Database Formal Definition

- Three layers of definition
  - **Domain**
  - **Relation Schema**; **relation state**
  - **Database schema**; **database state**

Values and NULLs in Tuples

- Atomic values
  - Each value in a tuple must be an atomic value
  - I.e., it is not divisible into components within the framework of the basic relational model
  - Composite and multivalued attributes are not allowed.
- **NULL values**
  - An important concept
  - Used to represent the values of attributes that
    - Exist but we don’t know
    - We don’t know whether exist
      - A student has a NULL for home phone (he doesn’t have, or he has but we don’t know.)
    - Do not apply to a tuple
      - A student has NULL for his office phones because he does not have office

(Relational) Model Constraints

- Rules imposed on the modeling of data to make database safer, more meaningful and more useful
- Three main categories
  - Inherent model-based constraints or implicit constraints:
    - Constraints that are inherent in the data model
    - E.g. Tuples cannot repeat in a relation in RDB
  - Schema-based constraints or explicit constraints
    - Constraints that can be directly expressed in schemas of the data model, typically by specifying them in the DDL (data definition language)
  - Application-based or semantic constraints or business rules
    - Constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs.

(Relational) Model Constraints

- (Additional) rules that we impose on our modeling of data to make database safer, more meaningful and more useful
- Three main categories
- Inherent model-based constraints or implicit constraints:
  - Constraints that are inherent in the data model
  - E.g. Tuples cannot repeat in a relation in RDB
- Schema-based constraints or explicit constraints
  - Constraints that can be directly expressed in schemas of the data model, typically by specifying them in the DDL (data definition language)
- Application-based or semantic constraints or business rules
  - Constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs.

Key Constraints (1)

- A relation is defined as a set of tuples
- Since all elements of a set must be distinct; hence, all tuples in a relation must also be distinct
  - ⇒ I.e., no two tuples can have the same combination of values for all their attributes.
- Usually, there may be **subsets of attributes** in a relation schema R with the property that no two tuples in any relation state r(R) should have the same combination of values for these attributes.
  - Denote one such **subset of attributes** by **SK**
  - ⇒ For any two distinct tuples t1 and t2 in a relation state rof R, we have the constraint that: $t1[SK] \neq t2[SK]$

Key Constraints (2)

- Any such set of attributes SK is called a superkey of the relation schema R.
  - A superkey SK specifies a uniqueness constraint that no two distinct tuples in any state r(R) can have the same value for SK.
  - Every relation has **at least one default superkey** — the set of all its attributes
- Definition: a key K of a relation schema R
  - Is a superkey of R
  - Removing any attribute A from K, the rest of attribute set K' is no longer a superkey of R
- Properties of key
  - A key is also a superkey, but not vice versa.
  - Any superkey formed from one single attribute is always also a key

Key Constraints (3)

- In general, a relation schema may have more than one key. In this case, each of the keys is called a candidate key.
- For example
  - A relation for CAR may have two candidate keys: License_number and Engine_serial_number
  - A relation for student may have two candidate keys: student_ID and National_ID
- One of the candidate keys is designated as the primary key of the relation
  - The values of primary key are used to identify tuples in the relation.
  - Convention: the attributes that form the primary key of a relation schema are underlined
- Choice of one candidate keys to become primary key is somewhat arbitrary
  - Usually better to choose a primary key with a single attribute or a small number of attributes
  - The other candidate keys are designated as unique keys, and are not underlined

Constraints on NULL

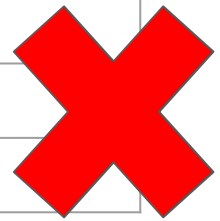
- Specifies whether NULL values are or are not permitted for a particular attribute
  - For example, if every STUDENT tuple must have a known, valid value for the Name attribute, then Name of STUDENT is constrained to be NOT NULL.

Entity Integrity Constraints

- The entity integrity constraint states that no primary key value can be NULL.
  - Because the primary key value is used to identify individual tuples in a relation, having NULL values for the primary key implies that we cannot identify some tuples.
  - For example, if two or more tuples had NULL for their primary keys,we may not be able to distinguish them if we try to reference them from other relations.

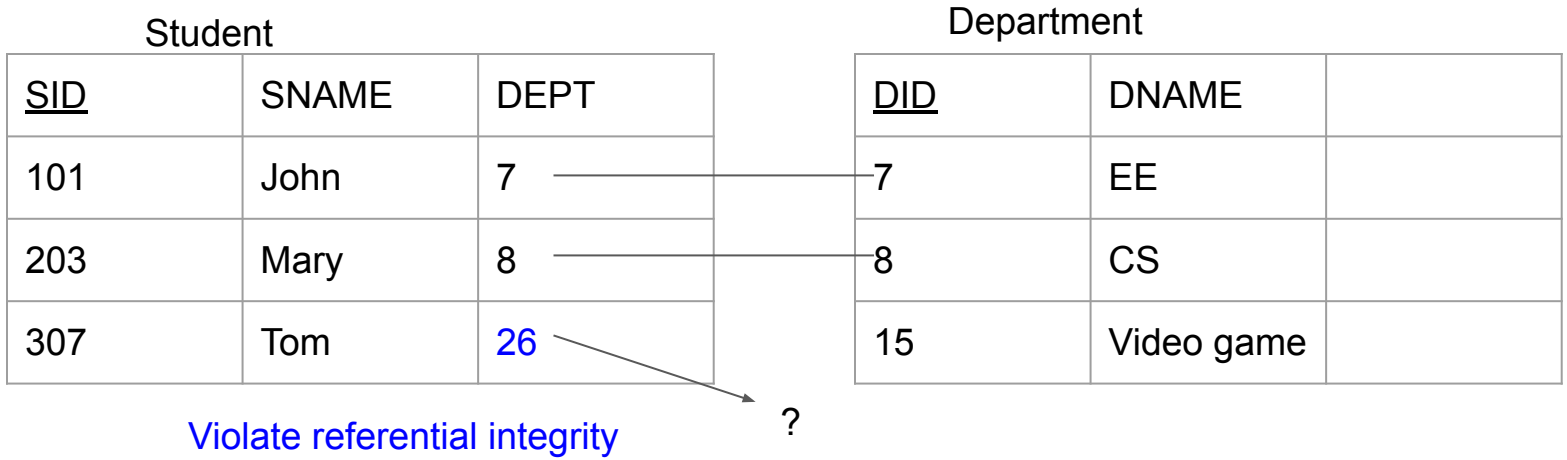
EID	ENAME
101	John
102	Mary

EID	ENAME
101	John
NULL	Mary



Referential Integrity Constraints

- The referential integrity constraint is specified between two relations
- is used to maintain the consistency among tuples in the two relations.
- Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.



Transaction

- A transaction
  - Is a unit of work that includes **several database operations**
    - such as reading from the database, or insertions, deletions, or updates to the database.
  - At the end of a transaction
    - It must leave the database in a valid state
    - Must satisfies all the constraints specified on the database schema.
    - All operations either all succeed together, or all failed together
      - Cannot let it happen ⇒ some succeed, some fail
    - These operations forms an atomic unit of work against the database.
  - For example, a transaction to apply a bank withdrawal will typically read the user account record, check if there is a sufficient balance,and then update the record by the withdrawal amount.

SQL Overview

Terminology about Database Sublanguages

Broadly speaking

- DDL: data definition language
- DML: data manipulation language

Divide into details (informally, usually by commercial systems)

- DDL: data definition language
- SDL: storage definition language
- DCL: data control language
- DQL: data query language
- DML: data manipulation language

What is SQL?

- E.g.
  SELECT * FROM department
  WHERE name = 'EE';

- A language for defining your relational database model (DDL)
- A language to access relational data (DML)
- An Interface to Relational Database
- An important tool for DBMS
- A declarative language
- A language with “life”
- Borrowed from and extended to many applications beyond RDB

- We will present the “principles” of SQL
- SQL you encounter in actual systems in the future will all have small variations.

SQL DDL
(Data Definition Language)

Create Table

- **CREATE TABLE** PROJECT
  (Pname VARCHAR(15) **NOT NULL**,
  Pnumber INT **NOT NULL**,
  Plocation VARCHAR(15),
  Dnum INT **NOT NULL**,
  PRIMARY KEY (Pnumber),
  UNIQUE (Pname),
  FOREIGN KEY (Dnum)
  REFERENCES DEPARTMENT(Dnumber));

Specifying Key Constraints

- **PRIMARY KEY** clause: specify primary key of a relation
  - If a primary key has a single attribute, the clause can follow the attribute directly.
  - For example,the primary key of PROJECT can be specified as follows Dnumber INT PRIMARY KEY;
- **UNIQUE** clause: specify secondary key of a relation
  - The UNIQUE clause can also be specified directly for a secondary key if the secondary key is a single attribute,
  - For example: Pname VARCHAR(15) UNIQUE;

Specifying Key Constraints (2)

- **Example**
  CREATE TABLE PROJECT
  (Pname VARCHAR(15) **NOT NULL**,
  Pnumber INT **NOT NULL**,
  Plocation VARCHAR(15) DEFAULT 'TAIPEI',
  Dnum INT **NOT NULL CHECK** (Dnum > 0
  AND Dnum <21),
  PRIMARY KEY (Pnumber),
  UNIQUE (Pname),
  FOREIGN KEY (Dnum)
  REFERENCES DEPARTMENT(Dnumber));

- **Example**
  CREATE TABLE PROJECT
  (Pname VARCHAR(15) **UNIQUE NOT NULL**,
  Pnumber INT **PRIMARY KEY NOT NULL**,
  Plocation VARCHAR(15) DEFAULT 'TAIPEI',
  Dnum INT **NOT NULL CHECK** (Dnum > 0
  AND Dnum <21),
  FOREIGN KEY (Dnum)
  REFERENCES DEPARTMENT(Dnumber));

Specifying Referential Integrity Constraints

- FOREIGN KEY clause
- **Example**

```
CREATE TABLE PROJECT
( Pname VARCHAR(15) NOT NULL,
Pnumber INT NOT NULL,
Plocation VARCHAR(15) DEFAULT 'TAIPEI',
Dnum INT NOT NULL CHECK (Dnum > 0 AND Dnum <21),
PRIMARY KEY (Pnumber),
UNIQUE (Pname),
FOREIGN KEY (Dnum)
REFERENCES DEPARTMENT(Dnumber));
```

What does this constraint really regulate?

In what situation will this command fails?

Does it affect other commands?

Schema Changing: Drop

- Drop
  - Can drop database, schema, catalog
  - Can drop table
  - Restrict or cascade clause: decide what to do with affected objects
- DROP DATABASE university (mySQL)
- DROP SCHEMA university
  - In MySQL, you can say DROP DATABASE ...
- DROP SCHEMA university CASCADE
- DROP TABLE departments RESTRICT
- DROP TABLE departments CASCADE

Schema Changing: Alter

- ALTER {DATABSE|SCHEMA} db_name alter_specification …;
- ALTER TABLE: change the details of a table definition
  - Can add, drop column, constraint, index
  - Can rename column or change the definition of column
- ALTER TABLE students **ADD COLUMN** grade DECIMAL(5.2);
- ALTER TABLE students **DROP COLUMN** grade CASCADE;
  - We can specify CASCADE or RESTRICT when drop column
- ALTER TABLE students ADD COLUMN grade int DEFAULT 1.0;
- ALTER TABLE students **ALTER COLUMN** grade SET DEFAULT 2.0;
- ALTER TABLE students ALTER COLUMN grade DROP DEFAULT;

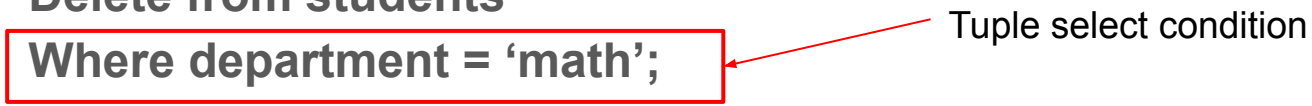
SQL DML
(Data Manipulation Language)

Insert

- Three kinds of usage
  - Specific all values
  - Specific attribute names and values
  - Get value from a select clause
- Create table students (
  ID int, name char(20), department char(20));
- **Insert into students values (1, “John smith”, “Math”);**
- **Insert into students (ID, name) values (2, “Peter Chen”);**
- **Insert into students**
  Select HID, Hname, Wishdept From highschool_students
  Where status = ‘selected’;

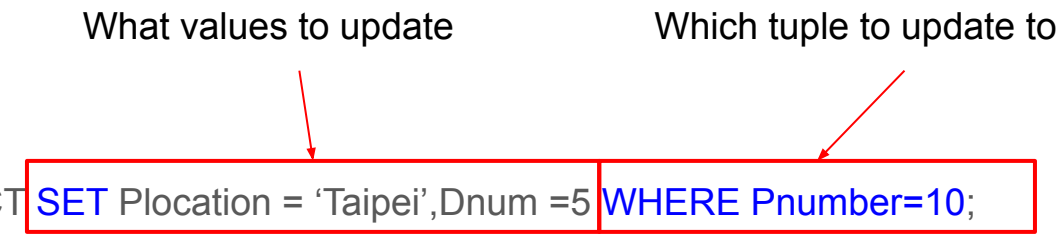
Delete

- Two types of usage
  - With conditions
  - Without condition
- Create table students (
  ID int, name char(20), department char(20));
- **Delete from students**
  Where department = ‘math’;
- **Delete from students;**
  - Delete all rows in the table 'students’ ⇒ Be careful
  - But the schema of ‘students’ still remains



Update

- Modify attribute values of one or more selected tuples
  - Specify what values to update, and which tuple to update to
- Update tablename set attribute='value', attribute='value'... where condition
- Can also set to NULL and set to default



- Examples
  - UPDATE PROJECT SET Plocation = 'Taipei',Dnum =5 WHERE Pnumber=10;
  - UPDATE EMPLOYEE SET Salary = Salary * 1.1 WHERE Dno = 5;
  - UPDATE EMPLOYEE SET Salary = DEFAULT WHERE Dno = 5;
  - UPDATE EMPLOYEE SET Salary = NULL WHERE Dno = 5;

Select (Basic)

- Most important command in SQL
- The way to execute query and get answers!
- Basic format
  SELECT <attribute list>
  FROM <table list>
  WHERE <condition>;
- Note: when there are multiple tables in the <table list>, it means some kind of join operation
- <condition> typically contains atomic condition expressions connected by AND, OR, NOT

Select (2)

- Create table students (
  ID int, name char(20), deptID int);
- Create table departments (
  DID int, Dname char(20));
- Select ID, name
  From students
  Where deptID = 3;
- Select ID, name
  From students, departments
  Where Dname=’EE’’ and deptID=did;

Select (2-1)

- Select ID, name
  From students, departments
  Where Dname=’EE’’ and DEPTID = DID;

Student: Referencing relation

Department: Referenced relation

ID	NAME	DEPTID
101	John	7
203	Mary	8
307	Tom	NULL

DID	DNAME	
7	EE	
8	CS	
15	Video game	

Referential integrity holds between Student and Department now

Transaction

- Basic syntax:
  - START TRANSACTION;
    <sql commands>...
    <sql commands>...
    <sql commands>...
    COMMIT (or ROLLBACK)
  - SET autocommit = {0|1}
    - Default is ‘1’
- START TRANSACTION;
  SELECT @A:=SUM(salary) FROM table1 WHERE type=1;
  UPDATE table2 SET summary=@A WHERE type=1;
  COMMIT;