LECTURE 2:
RELATIONAL DATABASES

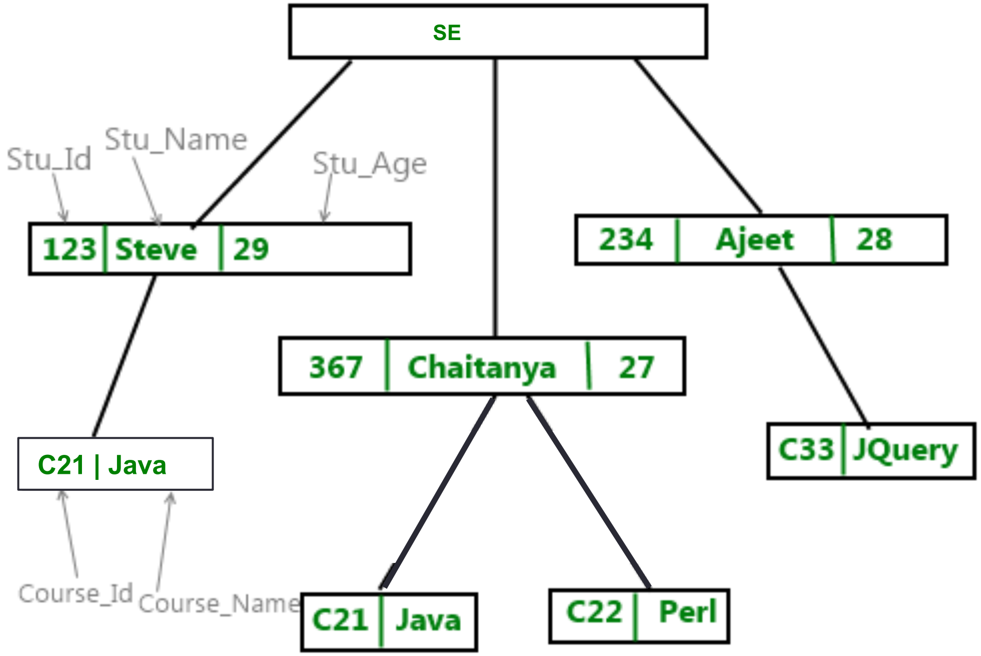
COMP2004J: Databases and Information Systems

Dr. Ruihai Dong (ruihai.dong@ucd.ie)

UCD School of Computer Science

Beijing-Dublin International College

Hierarchical Based Database (Review)

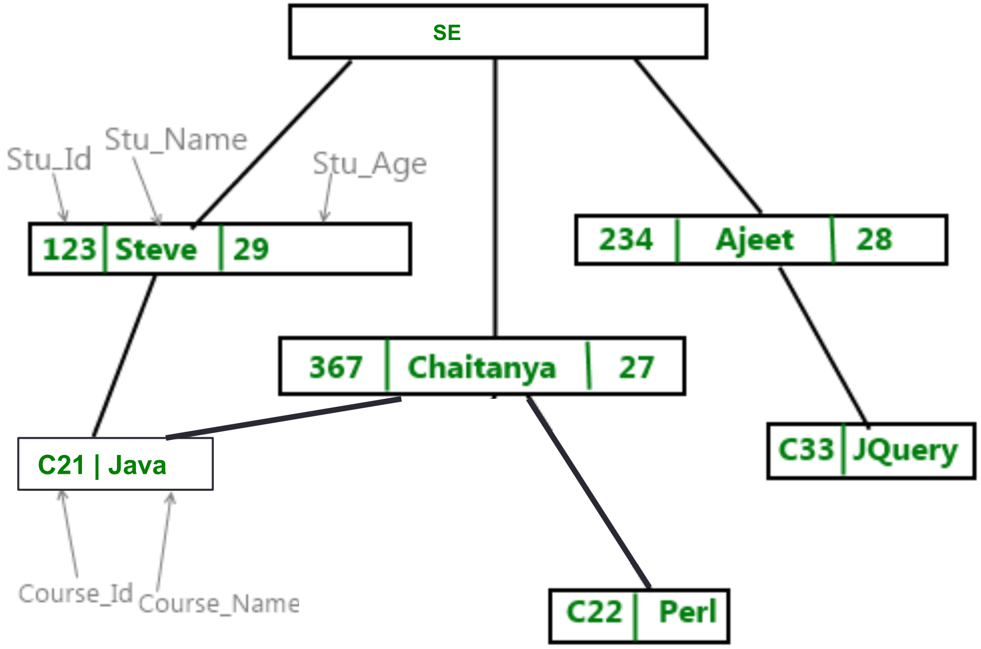


The layout is highly efficient for operations that “drill down,” so queries such as “how many courses does Chaitanya choose?” are easy to answer.

However, non-hierarchical queries are difficult to express and very costly to evaluate: queries such as “how many students do study Java?”

The hierarchical model is also prone to redundancy.

Network Database System (Review)



The network model generalizes the hierarchical model to represent relationships as directed graphs rather than trees. Doing so improves efficiency significantly: an application can easily add connections between records in order to accelerate important queries.

This model also has some problems, the main problem being that you need to be a database expert to use this database successfully. It is very difficult for the general public to use.

Relational Model

- The relational model can be seen as having three aspects

- Structural aspect
  - All data is held in tables.
  - Relationships between data are not explicitly stored.

- Integrity aspect
  - All tables satisfy integrity constraints (what can be stored).

- Manipulative aspect
  - Operators derive new tables from existing tables.

Relational Concepts (Review)

student

StudentNo	FirstName	LastName	Year	Major
1312345	Lina	Xu	2013	Finance
1318999	Wan	Wan	2013	Software Engineering
1218985	Ning	Cao	2012	Internet of Things

course

CourseCode	Title	Teacher
COMP2007J	Databases	Sean Russell
COMP2003J	Data Structures	David Lillis

exam

StudentNo	Grade	CourseCode
1312345	A-	COMP2007J
1218985	B+	COMP2003J

Data is represented as a collection of **relations**

Relational Concepts (Review)

- Each relation is **table** of values
- Each table consists of **rows** and **columns**

Student

StudentNo	FirstName	LastName	Year	Major
1312345	Lina	Xu	2013	Finance
1318999	Jessi	Wan	2013	Software Engineering
1218985	Ning	Cao	2012	Internet of Things

Database Structure

student

StudentNo	FirstName	LastName	Year	Major
1312345	Lina	Xu	2013	Finance
1318999	Wan	Wan	2013	Software Engineering
1218985	Ning	Cao	2012	Internet of Things

course

CourseCode	Title	Teacher
COMP2007J	Databases	Sean Russell
COMP2003J	Data Structures	David Lillis

exam

StudentNo	Grade	CourseCode
1312345	A-	COMP2007J
1218985	B+	COMP2003J

Database Structure

- Generally we describe the structure of a database in terms of its **relations** and **attributes**

- The structure of the relations from the previous slide are

```
student (StudentNo, FirstName, LastName, Year, Major)

course (CourseCode, Title, Teacher)

exam (StudentNo, grade, CourseCode)
```

Database Integrity (Correctness)

- Database integrity is about rules that can be applied to the data

- These rules are called **integrity constraints**
  - Domain integrity
  - Entity integrity
  - Referential integrity

Domain Integrity

- Domain integrity specifies that all columns in a relational database be declared upon a defined domain.

StudentNo	FirstName	LastName	Year	Gpa
1312345	Lina	Xu	2013	3.23
1318999	Wan	Wan	2013	2.89
1218985	Ning	Cao	2012	3.94

- For example,
  - The **type** of data stored in **FirstNmae** is text
  - The **type** of data stored in **Year** is an integer
  - The **type** of data stored in **Gpa** is a real number

Entity Integrity

- Entity integrity states that every table should have a primary key and the column or columns chosen to be the primary be unique and not null.

- A **primary key** is an attribute or a set of attributes that **uniquely identifies** records within a relation
  - This is required because a relation can contain no duplicates

Primary Keys

- In most modern databases this attribute is added in order to make sure the each record is **unique**
  - For example it is very likely that two students could share the same first name and family name
  - This is why all students are identified by a number

StudentNo	FirstName	LastName	Year	Gpa
1312345	Lina	Xu	2013	3.23
1318999	Wan	Wan	2013	2.89
1218985	Ning	Cao	2012	3.94
1213333	Ning	Cao	2012	1.25

Primary Keys

```
student (StudentNo, FirstName, LastName, Year, Major)
course (CourseCode, Title, Teacher)
exam (StudentNo, grade, CourseCode)
```

- For each of the tables, the primary key is shown as underlined
  - This means the **StudentNo** attribute is the key for the **student** relation
- A number of Attributes can be used to together as the key of a table
  - This means the combination of **StudentNo** and **CourseCode** is the primary key for the **exam** relation
- Primary keys are shown by underlining the attributes

Combined Keys

- When using a combined primary key we can have duplicates of **part** of the key but not all of it

```
exam (StudentNo, grade, CourseCode)
```

- In this example the same student number can be repeated and so can the same coursecode

- But you cannot have the same combination twice

Combined Keys

exam

StudentNo	Grade	CourseCode
1312345	A-	COMP2007J
1218985	B+	COMP2007J
1312345	A+	COMP2003J
1312345	A+	COMP2002J
1218985	C-	COMP2002J

We can have duplicates of **part** of the combined key

Combined Keys

exam

StudentNo	Grade	CourseCode
1312345	A-	COMP2007J
1218985	B+	COMP2007J
1312345	A+	COMP2003J
1312345	A+	COMP2002J
1218985	C-	COMP2002J
1312345	A	COMP2007J

Error!

But you cannot have the same combination twice

Foreign Keys

student

StudentNo	FirstName	LastName	Year	Major
1312345	Lina	Xu	2013	Finance
1318999	Wan	Wan	2013	Software Engineering
1218985	Ning	Cao	2012	Internet of Things

exam

StudentNo	Grade	CourseCode
1312345	A-	COMP2007J
1218985	B+	COMP2003J

course

CourseCode	Title	Teacher
COMP2007J	Databases	Sean Russell
COMP2003J	Data Structures	David Lillis

A *foreign key* is an attribute or a set of attributes in a relation that is a **primary key in another relation**.

Foreign keys are used to link data together in different relations.

Foreign Keys

student (StudentNo, FirstName, LastName, Year, Major)
course (CourseCode, Title, Teacher)
exam (*StudentNo*, grade, *CourseCode*)

- Here ***StudentNo*** and ***CourseCode*** in the exam relation are both foreign keys
  - ***StudentNo*** matches to the primary key of the student relation
  - ***CourseCode*** matches to the primary key of the course relation

- There is no physical link between the tables, only values which can be used to access data in other tables

- Foreign keys are usually shown in *italics*

Referential Integrity

- Referential integrity concerns the concept of a foreign key. When using foreign keys we combine data together only where the foreign key in our table matches the primary key in another table.
- Referential integrity specified between two relations and is used to maintain the consistency among rows in the two tables. Informally, the referential integrity constrain states that a row in one table that refers to another table must refer to an existing row in that table.

Referential Integrity

student

StudentNo	FirstName	LastName	Year	Major
1312345	Lina	Xu	2013	Finance
1318999	Wan	Wan	2013	Software Engineering
1218985	Ning	Cao	2012	Internet of Things

exam

StudentNo	Grade	CourseCode
1312345	A-	COMP2007J
1218985	B+	COMP2003J

course

CourseCode	Title	Teacher
COMP2007J	Databases	Sean Russell
COMP2003J	Data Structures	David Lillis

The data stored in the **StudentNo** attribute in **exam** **must match one** of the entries in **StudentNo** attribute in **student**

The data stored in the **CourseCode** attribute in **exam** **must match one** of the entries in **CourseCode** attribute in **course**

Referential Integrity

course

CourseCode	Title	Teacher
COMP2007J	Databases	David Lillis
COMP2003J	Data Structures 2	David Lillis
COMP2002J	Data Structures 1	Sean Russell

exam

StudentNo	Grade	CourseCode
1312345	A-	COMP2007J
1218985	B+	COMP2007J
1312345	A+	COMP2003J
1312345	A+	COMP2002J
1218985	C-	COMP2002J
1218985	B	COMP4001J

Because this row contains a `CourseCode` that does not exist in the `course` relation, it is said to **violate the foreign key constraint (Referential Integrity).**

OPERATORS

Operators

- There are three main categories of operators for relational databases

- Project
  - Choose which columns we want

- Restrict (Select)
  - Choose which rows we want

- Join
  - Combine two or more tables

Operators: Project

- The project operator takes **chosen** attributes from a relation

- In SQL this is performed using the SELECT command:
  - `SELECT empno, name FROM employee;`
  - `SELECT name, address FROM student;`

- This type of query **returns every record**, but only some of the attributes of each record

- A Projection is a **vertical** cut of the table

Operators: Project

students

studentno	studentname	major	year
1312345	Xu Lina	Finance	2013
1318999	Jie Wan	Software Engineering	2013
1218985	Cao Ning	Internet of Things	2012

```
SELECT studentno, studentname FROM student;
```

Note that SQL commands end with a semicolon.

Note that by convention we use uppercase letters for SQL commands, and lowercase when naming things.

students

studentno	studentname
1312345	Xu Lina
1318999	Jie Wan
1218985	Cao Ning

Operators: Project

```
SELECT studentno, studentname FROM student;
```

- Between the SELECT and the FROM, we name the **attributes** that we are interested in projecting.
- Every record in the student relation is returned, but not all of the attributes is shown: only the attributes that we requested.
- If we do not want to use projection in SQL we place a * between SELECT and FROM (this will return all attributes)

```
    SELECT * FROM student;
```

Operators: Restrict

- The restrict operator takes **chosen records** from a relation

- In SQL, to perform a **restrict** operation, we add to the SELECT command:

```
SELECT * FROM exam WHERE grade='A+';
SELECT * FROM student WHERE major='Finance';
```

- This type of operator is returns a **subset of the data**

- A restriction is a **horizontal** cut of the table

Operators: Restrict

student

studentno	studentname	major	year
1312345	Xu Lina	Finance	2013
1318999	Jie Wan	Software Engineering	2013
1218985	Cao Ning	Internet of Things	2012

```
SELECT * FROM student WHERE year = 2012;
```

student

studentno	studentname	major	year
1218985	Cao Ning	Internet of Things	2012

Operators: Restrict

- `SELECT * FROM student WHERE year = 2012;`

- The **WHERE** clause of a SELECT statement indicates the records that we are interested in.

- **Every record** in the students relation is individually checked to see if “year=2012" for that row.
  - Rows where “year=2012" is true are returned in the output.
  - Other rows are not.

Notice that we do not put quotes around numeric data but we put single-quotes around non-numeric values:

```
SELECT * FROM student WHERE year=2012;
SELECT * FROM student WHERE name='Jie Wan';
```

Combining Restriction and Projection

- The **restrict** and **project** operations can be combined into one SELECT command if we wish:

```
SELECT studentname, major FROM student WHERE
year=2013;
```

  - We **project** the table so we only get the **studentname** and **major** attributes.
  - We **restrict** our output to those students that began their courses in 2013.

Operators: Restrict

student

studentno	studentname	major	year
1312345	Xu Lina	Finance	2013
1318999	Jie Wan	Software Engineering	2013
1218985	Cao Ning	Internet of Things	2012

```
SELECT studentname, major FROM student
WHERE year=2013;
```

student

studentname	major
Xu Lina	Finance
Jie Wan	Software Engineering

Sometimes I need to put a SQL query on two lines because the full command won’t fit in a PPT slide.

It’s best to keep each query on the same line.

Quiz

StudentNo	FirstName	LastName	Year	Gpa
1312345	Lina	Xu	2013	3.23
1318999	Wan	Wan	2013	2.89
1218985	Ning	Cao	2012	3.94
1313333	Ning	Cao	2013	1.25

```
SELECT StudentNo, Gpa FROM student WHERE FirstName=‘Ning’
AND LastName=‘CAO’ AND year=2013;
```

Operators: Join

- Join takes records from two relations based on some **join condition**

```
SELECT studentname, coursecode, grade FROM student, exam
WHERE student.studentno = exam.studentno;
```

- The join condition here is the clause
  - `student.studentno = exam.studentno`

- Here '.' is called the dot membership operator
  - **`student.studentno`** refers to the `studentno` attribute in the `student` relation
  - **`exam.studentno`** refers to the `studentno` attribute in the `exam` relation

Database Structure

student

studentno	studentname	major	year
1312345	Xu Lina	Finance	2013
1318999	Jie Wan	Software Engineering	2013
1218985	Cao Ning	Internet of Things	2012

course

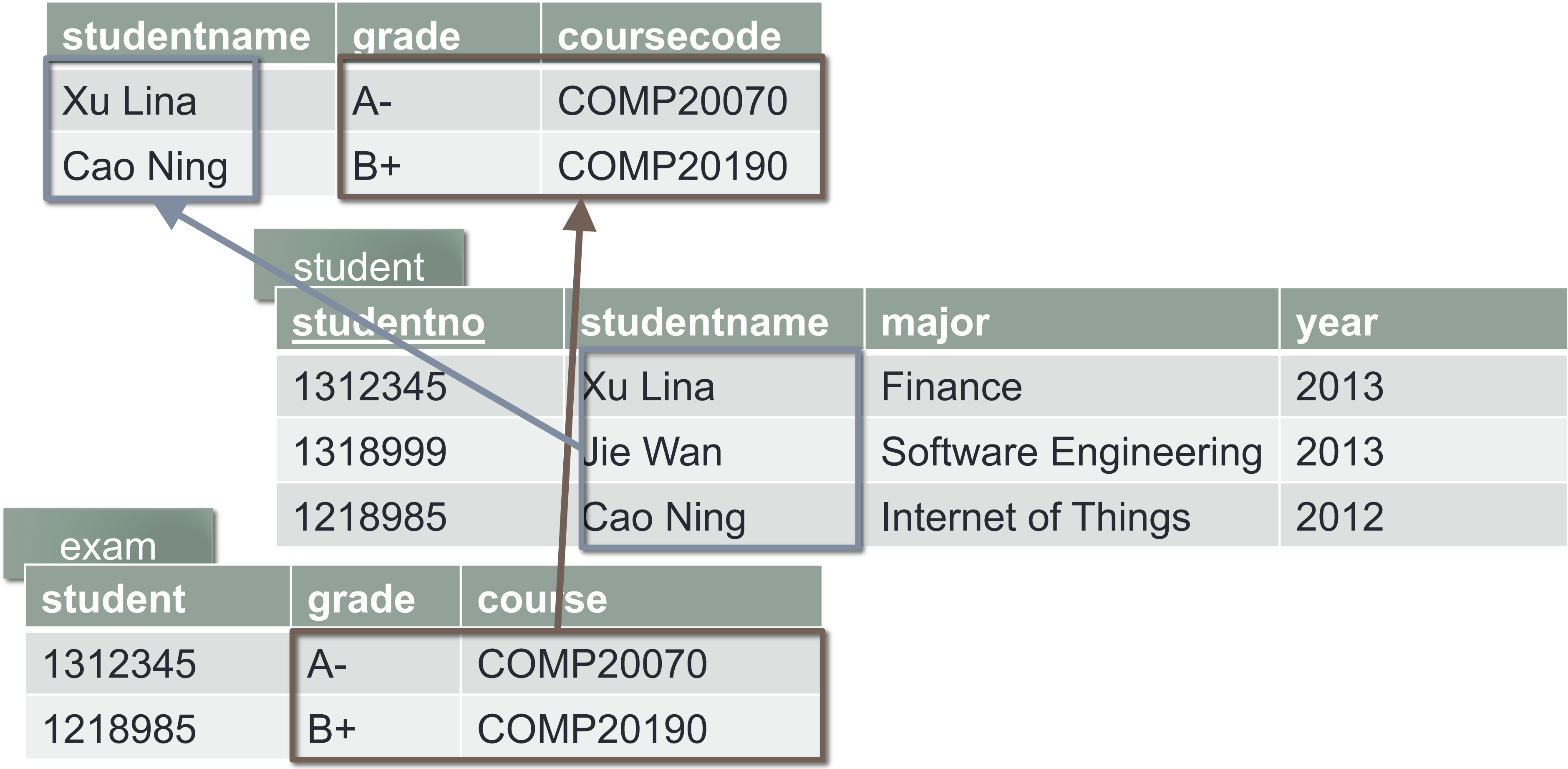
code	title	teacher
COMP20070	Databases	Sean Russell
COMP10110	Programming	Rem Collier

exam

studentno	grade	*coursecode*
1312345	A-	COMP20070
1218985	B+	COMP20190

Operators: Join

```
SELECT studentname, grade, coursecode FROM student,
exam WHERE student.studentno = exam.studentno;
```



studentname	grade	coursecode
Xu Lina	A-	COMP20070
Cao Ning	B+	COMP20190

student

studentno	studentname	major	year
1312345	Xu Lina	Finance	2013
1318999	Jie Wan	Software Engineering	2013
1218985	Cao Ning	Internet of Things	2012

exam

student	grade	course
1312345	A-	COMP20070
1218985	B+	COMP20190

Closure

- The fact that the result of any operation is another relation is known as the **closure** property

- It means that the output from one operation can be the input to another operation

- `SELECT studentname, major FROM student;`

- The records returned by this query obey all the rules of the relational model
  - This means that we can perform **another query** on the result

Closure

- The Closure property means it is possible to write **nested** expressions.
    - This means one query inside another.

- When we say the output of an operation is a relation, we mean that logically it is a relation and so is available for the next operation.

- How it is actually stored, or not, is a matter for the DBMS and not something we need to worry about.