Databases and Info Systems

Relational Databases

Dr. Seán Russell
sean.russell@ucd.ie,

School of Computer Science,
University College Dublin

February 19, 2020

Table of Contents

1. Relational Model
2. Structure
3. Integrity
4. Manipulating Relational Databases
5. Closure

Relational Model

- There are three aspects of the relational model that we must understand
  - How it is **structured**
  - How it maintains **integrity**
  - How it can be **manipulated**

Table of Contents

1. Relational Model
2. Structure
3. Integrity
4. Manipulating Relational Databases
5. Closure

Database Structure

- Data in the relational models is stored in a collection of **relations**
  - We will use the term **table** when discussing relations
- The relationships between tables is not explicitly stored

Database Structure Example

students

student_num	name	major	year_of_entry
17206777	Sean Russell	SE	2017
18205333	David Lillis	IOT	2016
16205777	Brett Becker	EIE	2016

modules

module_code	title	teacher
COMP2013J	Database	Sean Russell
COMP2011J	OOP	Sean Russell
COMP2003J	Data Structures	David Lillis

Structure Example

- The structure of a database is described in terms of **relations** and **attributes**
- We use this format to describe them
  - `relation_name ( attribute_name1, a_name2, a_name3 )`
- The relations from our example are
  - `students( student_num, name, major, year_of_entry )`
  - `modules( module_code, title, teacher )`
  - `results( student_num, grade, module_code )`

Connections

- We can see that data in some of these tables are related
- But this is not recorded in the structure of the database
- We have to create these connections when we are searching for data

Table of Contents

1. Relational Model
2. Structure
3. Integrity
   - Keys
4. Manipulating Relational Databases
5. Closure

Database Integrity

- Database integrity is the correctness of the data
- Integrity is ensured using a number of constraints (rules)
    - Domain Integrity Constraint
    - Entity Integrity Constraint
    - Referential Integrity Constraint

Domain Integrity Constraint

- The domain integrity constraint states that all columns in a relational database are in a defined **domain**
- This is similar to data types in programming
  - E.g. `student_num` can only allow integer values
  - E.g. `module_code` can only allow text values

Entity Integrity Constraint

- The entity integrity is concerned with the concept of primary keys
- The rule states that every table must have its own **primary key**
- The value of this attribute must be **unique** and **not null** for every row

Referential Integrity Constraint

- The referential integrity constraint is the concept of foreign keys
- The rule states that the foreign key value can be in two states.
  - The first state is that the **foreign key** value would refer to a **primary key** value of another table
  - The second state is that it can be **null**
- Being null could simply mean that there are no relationships, or that the relationship is unknown

Keys

- The entity and referential integrity constraints both referred to **keys**
- A key is an attribute or set of attributes that **uniquely identifies** rows in a table
  - Remember no duplicate rows are allowed

Primary Key

- Any key that uniquely identifies a relation is called a **candidate key**
- From the candidate keys a single **primary key** is chosen to identify the relation
- In many modern databases, this attribute is added to make sure each record is **unique**
  - This is why you have a student number
  - Many students may have the same name

Keys

- Primary keys are shown by underlining the attributes
- Multiple attributes can be used together as the primary key
- Our example redone would be:

Example

```
students( student_num, name, major, year_of_entry )
modules( module_code, title, teacher )
results( student_num, grade, module_code )
```

Combined Keys

- When using a combined primary key we can have duplicates of **part** of the key but not all of it
  - `results( student_num, grade, module_code )` (student_num and module_code underlined)
- In this example the same `student_num` can be repeated and so can the same `module_code`
- But you cannot have the same combination twice

Foreign Keys

- A foreign key is an attribute or set of attributes in a relation that is a key in another relation
- Foreign keys are used to link data together in different relations
- When using foreign keys, we combine data together only where the foreign key in our table matches the primary key in another table

Foreign Keys

Example

students(<u>student_num</u>, name, major, year_of_entry)
modules(<u>module_code</u>, title, teacher)
results(<u>student_num</u>, grade, <u>module_code</u>)

- Here `student_num` and `module_code` in the results relation are both **foreign** keys
  - `student_num` matches to the primary key of students
  - `module_code` matches to the primary key of modules
- Foreign keys do not need the same name as the primary key
- But it is common to have the same or similar names to help identify foreign keys

Foreign Keys Example

module_code	title	teacher
COMP2013J	Databases	Sean Russell
COMP2014J	Data Structures 2	David Lillis
COMP2011J	OOP	Sean Russell

student_num	grade	module_code
1312345	A-	COMP2013J
1218985	B+	COMP2011J
1312345	A+	COMP2011J
~~1412345~~	~~C+~~	~~COMP2015J~~

The last row contains a module code that does not exist (violates the referential integrity constraint)

Table of Contents

1. Relational Model
2. Structure
3. Integrity
4. Manipulating Relational Databases
   - Example Data
   - Project Operator
   - Restrict Operator
   - Join Operator
5. Closure

Operators

- There are three main categories of operators we can use to manipulate relational databases

  Project - choose which columns/attributes we want to see

  Restrict - choose which rows/tuples we want to see (sometimes called select)

  Join - combine two or more tables

students Relation

student_num	name	major	year_of_entry
19206555	Jordan Larmour	IOT	2019
18206123	Jordan Murphy	SE	2017
15205999	Jamie Heaslip	FIN	2015
16206456	Gary Ringrose	EIE	2017
18205555	Gavin Henshaw	SE	2018

courses Relation

<u>module_code</u>	title	teacher
COMP1002J	Intro to Programming 2	John Dunnion
COMP1004J	Intro to Program Construction 1	Sean Russell
COMP2013J	Databases and Information Systems	Sean Russell
COMP2014J	Data Structures and Algorithms 2	David Lillis

results Relation

student_num	grade	module_code
18206123	A+	COMP2014J
18205555	C	COMP1004J
19206555	FM	COMP1002J
16206456	B	COMP2014J

Projection

- The **project** operator allows you to choose which attributes/columns of a relation you want to see
- **All** rows in the relation will be returned
- This is performing a vertical cut of your relation/table

SQL Projection

- Getting data from an SQL database, requires the use of the `SELECT` command
- The syntax of `SELECT` is:
  `SELECT projection FROM relation;`
  - E.g. `SELECT name, major FROM students;`
- The attributes that we list are the only ones returned in the result
- If we want all of the attributes we use * instead
  - E.g. `SELECT * FROM courses;`

Example of Projection

students relation

student_num	name	major	year_of_entry
19206555	Jordan Larmour	IOT	2019
18206123	Jordan Murphy	SE	2017
15205999	Jamie Heaslip	FIN	2015
16206456	Gary Ringrose	EIE	2017
18205555	Gavin Henshaw	SE	2018

```
SELECT name, major FROM students;
```

name	major
Jordan Larmour	IOT
Jordan Murphy	SE
Jamie Heaslip	FIN
Gary Ringrose	EIE
Gavin Henshaw	SE

Example of Projection

students relation

student_num	name	major	year_of_entry
19206555	Jordan Larmour	IOT	2019
18206123	Jordan Murphy	SE	2017
15205999	Jamie Heaslip	FIN	2015
16206456	Gary Ringrose	EIE	2017
18205555	Gavin Henshaw	SE	2018

SELECT * FROM students;

student_num	name	major	year_of_entry
19206555	Jordan Larmour	IOT	2019
18206123	Jordan Murphy	SE	2017
15205999	Jamie Heaslip	FIN	2015
16206456	Gary Ringrose	EIE	2017
18205555	Gavin Henshaw	SE	2018

Restriction

- The **restrict** operator allows you to choose which **tuples/rows** of a relation you want to see
- A **subset** of the rows in the relation will be returned
- This is performing a **horizontal** cut of your relation/table

SQL Restriction

- The restrict operator adds to the SELECT command
- The updated syntax is: `SELECT projection FROM relation WHERE restriction;`
    - E.g. `SELECT * FROM students WHERE major="SE";`
    - E.g. `SELECT * FROM results WHERE grade="A+";`

Example of Restriction

students relation

student_num	name	major	year_of_entry
19206555	Jordan Larmour	IOT	2019
18206123	Jordan Murphy	SE	2017
15205999	Jamie Heaslip	FIN	2015
16206456	Gary Ringrose	EIE	2017
18205555	Gavin Henshaw	SE	2018

```
SELECT * FROM students WHERE major="SE";
```

student_num	name	major	year_of_entry
18206123	Jordan Murphy	SE	2017
18205555	Gavin Henshaw	SE	2018

Example of Restriction

results relation

student_num	grade	module_code
18206123	A+	COMP2014J
18205555	C	COMP1004J
19206555	FM	COMP1002J
16206456	B	COMP2014J

```
SELECT * FROM results WHERE grade="A+";
```

student_num	grade	module_code
18206123	A+	COMP2014J

Where Clause

- The WHERE clause of a SELECT statement indicates the rows that we are interested in
- E.g. `SELECT * FROM students WHERE major="SE";`
- **Every** row in the students relation is individually checked to see if the value for major is "SE"
  - Rows where the value of major is "SE" are returned
  - Other rows are not

Combining Restriction and Projection

- The restrict and project operations can be combined into one `SELECT` command

- `SELECT name FROM students WHERE major="SE";`
  - First, we **restrict** to only the students studying Software Engineering (SE)
  - Then, we **project** the result to only see the names of the students

Example of Restriction and Projection

students relation

student_num	name	major	year_of_entry
19206555	Jordan Larmour	IOT	2019
18206123	Jordan Murphy	SE	2017
15205999	Jamie Heaslip	FIN	2015
16206456	Gary Ringrose	EIE	2017
18205555	Gavin Henshaw	SE	2018

```
SELECT name FROM students WHERE major="SE";
```

name
Jordan Murphy
Gavin Henshaw

Selecting from Multiple Tables

- Selecting from **two** tables gets the Cartesian product of the tables
  - It combines every row from the first table with every row from the second table
  - Often, this Cartesian product does not make logical sense and more is required to get a meaningful result

Cartesian Product Example 1

students relation

student_num	name	major	year_of_entry
19206555	Jordan Larmour	IOT	2019
18206123	Jordan Murphy	SE	2017
15205999	Jamie Heaslip	FIN	2015

results relation

student_num	grade	module_code
18206123	A+	COMP2014J
18205555	C	COMP1004J
19206555	FM	COMP1002J
16206456	B	COMP2014J

Cartesian Product Example 2

Cartesian Product of students and results

student_num	name	major	year_of_entry	student_num	grade	module_code
19206555	Jordan Larmour	IOT	2019	18206123	A+	COMP2014J
19206555	Jordan Larmour	IOT	2019	18205555	C	COMP1004J
19206555	Jordan Larmour	IOT	2019	19206555	FM	COMP1002J
19206555	Jordan Larmour	IOT	2019	16206456	B	COMP2014J
18206123	Jordan Murphy	SE	2017	18206123	A+	COMP2014J
18206123	Jordan Murphy	SE	2017	18205555	C	COMP1004J
18206123	Jordan Murphy	SE	2017	19206555	FM	COMP1002J
18206123	Jordan Murphy	SE	2017	16206456	B	COMP2014J
15205999	Jamie Heaslip	FIN	2015	18206123	A+	COMP2014J
15205999	Jamie Heaslip	FIN	2015	18205555	C	COMP1004J
15205999	Jamie Heaslip	FIN	2015	19206555	FM	COMP1002J
15205999	Jamie Heaslip	FIN	2015	16206456	B	COMP2014J

Join Operator

- The `JOIN` operator takes records from two relations based on some **join condition**

```
SELECT name, grade, module_code FROM students, results
    WHERE students.student_num = results.student_num;
```

- The join condition is the clause
`students.student_num = results.student_num`

Dot Membership Operator

- The . in the join condition is called the dot membership operator
- `students.student_num = results.student_num`
  - `students.student_num` refers to the `student_num` attribute in `students`
  - `results.student_num` refers to the `student_num` attribute in `results`

Join Condition

- The join condition only selects the rows in the cartesian product, where the two student number values are the same

Cartesian Product of students and results

s.student_num	name	major	year_of_entry	r.student_num	grade	module_code
19206555	Jordan Larmour	IOT	2019	18206123	A+	COMP2014J
19206555	Jordan Larmour	IOT	2019	18205555	C	COMP1004J
19206555	Jordan Larmour	IOT	2019	19206555	FM	COMP1002J
19206555	Jordan Larmour	IOT	2019	16206456	B	COMP2014J
18206123	Jordan Murphy	SE	2017	18206123	A+	COMP2014J
18206123	Jordan Murphy	SE	2017	18205555	C	COMP1004J
18206123	Jordan Murphy	SE	2017	19206555	FM	COMP1002J
18206123	Jordan Murphy	SE	2017	16206456	B	COMP2014J
15205999	Jamie Heaslip	FIN	2015	18206123	A+	COMP2014J
15205999	Jamie Heaslip	FIN	2015	18205555	C	COMP1004J
15205999	Jamie Heaslip	FIN	2015	19206555	FM	COMP1002J
15205999	Jamie Heaslip	FIN	2015	16206456	B	COMP2014J

Join Example

```sql
SELECT name, grade, module_code FROM students, results WHERE
    students.student_num = results.student_num;
```

Result

name	grade	module_code
Jordan Larmour	FM	COMP1002J
Jordan Murphy	A+	COMP2014J

Table of Contents

1 Relational Model

2 Structure

3 Integrity

4 Manipulating Relational Databases

5 Closure

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 name, major FROM students;`
- 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

Set Operations

- All of these operations are manipulating and producing relations containing data.
  - That is sets of data
- They do not work at the single record level.
- A relation of one row, or of no rows, is a valid relation and may arise because one row, or no rows, meet the criteria set by the operation
- Dealing with sets of data is a different approach to procedural data manipulation as with a procedural programming language