Robert Williamson
University of Toronto
2013

A brief intro to databases and SQL

A database is a way to efficiently store data in a way that is quick and easy to access. They are particularly useful for large datasets that are very interconnected. For example you may make a database to store information from a simulation you have created, or to store genetic information from a multiple treatment experiment.

This worksheet is intended to help you learn the basics of how databases group data and how one would implement them. The first thing to think about is what kinds of data you have and how those data are related to one another. We usually represent this as an entity relationship diagram, the full details of how to do this are outside the scope of the worksheet, but wikipedia gives a good summary (This youtube video is a really great intro to ER diagrams). This is basically a way to minimize repetitiveness in your data, and to break it down into the most basic sets of data and relationships. These data and relationships are represented by **tables** in your database.

Tables can be thought of as excel sheets, each table has a series of columns and rows. Each column stores a specific type of data (e.g. one column might store a name), and each row stores a unique set of data (e.g. the name, SIN, and department of a student). Tables can reference rows in other tables, for example in the *AdvisorDept* table below the SIN column refers to a SIN number from the *Advisor* column. Often tables will also have a **primary key** this is a value or set of values that makes a row unique in that table. For example in the *Advisor* column the SIN is the primary key, this means that no two rows in that table share the same SIN number. In the *StudentAdvisor* column our primary key would be made up of both the SIN and StudentNum, because no student could start in a given advisor's lab on multiple dates, but each SIN can have many StudentNums (i.e. advisors have more than one student) and each StudentNum can be associated with multiple SINs (i.e. some students are co-advised).

Databases are created and accessed using Structured Query Language (SQL). SQL commands vary based on specific database types, in this document I have used SQLite, a database software appropriate for use with relatively small datasets. Below is an example database that stores information on faculty, graduate students, departments, and the relationships between them. Each problem has an example SQL command, and then asks you to create a similar command for another task.

Robert Williamson
University of Toronto
2013

Student

StudentNum	Name	Degree
1	Robert	PhD
2	Ali	PhD
3	Arvid	Msc
4	Gavin	BS
5	John	BS
6	Stephen	PhD

Department

DeptID	Name
1	Evolution
2	Ecology
3	Oncology
4	Accounting
5	Math
6	Awesome

Advisor

SIN	Name
999999	Stephen
111111	Aneil
222222	Asher
333333	Helen
444444	Locke
555555	Don

AdvisorDept

SIN	DeptID
999999	2
111111	1
222222	2
333333	1
444444	4
555555	2

StudentAdvisor

SIN	StudentNum	StartDate
999999	1	1/1/1979
111111	2	3/23/2012
222222	3	4/7/2006
111111	4	8/12/1990
444444	1	8/15/2012
999999	5	5/12/2008
555555	6	6/17/2008

The above is an example database mapping students, advisors, and departments to each other.

1. Start by drawing the entity-relationship this database represents. (Normally you would do this step first, then make the tables, we're doing this a little backwards.)

Robert Williamson
University of Toronto
2013

2. The SQL statements below was used to create the Student table and StudentAdvisor table, write SQL statements to create the other tables.

CREATE TABLE Student(StudentNum integer primary key, Name string, Degree string);

CREATE TABLE StudentAdvisor(SIN integer, StudentNum integer, StartDate string, PRIMARY KEY(SIN, StudentNum), FOREIGN KEY (SIN) REFERENCES Advisor(SIN), FOREIGN KEY (StudentNum) REFERENCES Student(StudentNum));

3. The following SQL statement adds data to the database. Make SQL statements to add the rest of the data.

INSERT INTO Student (StudentNum, Name, Degree) VALUES (1, "Robert", "PhD");

INSERT INTO AdvisorDept (SIN, DeptID) VALUES (999999, 2);

Robert Williamson
University of Toronto
2013

4. For each of the following SQL statements how many entries will be retrieved? And what are they?

SELECT * FROM Advisor;
(Note: * means "print all the columns from this table")

SELECT StudentNum FROM Student WHERE Degree = "PhD";

SELECT * FROM Student WHERE StudentNum > 4;

SELECT * FROM Advisor WHERE Name = "Stephen" OR Name = "Aneil";

SELECT Name, DeptID FROM Advisor INNER JOIN AdvisorDept ON Advisor.SIN = AdvisorDept.SIN;
(Note: INNER JOIN takes two tables and smashes them together based on some column. So this command takes every row from *AdvisorDept* and fuses it with any row from *Advisor* where the SIN is the same in both tables.)

SELECT Advisor.Name, AdvisorDept.DeptID FROM Advisor INNER JOIN AdvisorDept ON Advisor.SIN = AdvisorDept.SIN WHERE AdvisorDept.DeptID = 2;

SELECT Advisor.Name, Department.Name FROM (Advisor LEFT JOIN AdvisorDept ON Advisor.SIN = AdvisorDept.SIN) INNER JOIN Department ON AdvisorDept.DeptID = Department.DeptID;

5. Write a SQL statement to accomplish each of the following tasks.

Robert Williamson
University of Toronto
2013

Select all Advisors who teach in Department #4

Select all of the students who's advisors teach in the Accounting department.

Select all students who are either Helen's OR Asher's students.

Select all students who started after 2008.

Select all advisors who have a student who is either doing a BS or a MSc.