Relational Databases, SQL

Ahmet Sacan

Databases and SQL

- Lecture Goals
  - Create a mirna-mrna targeting database from text file(s).
  - Query the database table(s) to find the genes that are predicted to be targeted by a mirna.

What you'll need

- ~~Firefox, SQLite plugin~~
- SqliteStudio

- Mirdb ~~and Targetscan~~ databases

DB Definitions

- Relational database: collection of *tables* (also called *relations*)
- Database Management System (DBMS): a software that is used to create, access, and maintain a database; e.g., sqlite, mysql, postgresql, oracle, sql server.
- Table: Collection of rows (also called *tuples* or *records*).
- Each row in a table contains a set of columns (also called *fields* or *attributes*).
- Each column has a type:
  - Text: VARCHAR(20)
  - Integer: INTEGER
  - Floating-point: FLOAT, DOUBLE
  - Date/time DATE, TIME, DATETIME
- Primary key: provides a unique identifier for each row (optional).
- Schema: the structure of the database tables
  - The table name
  - The names and types of its columns
  - Various optional additional information (defaults, constraints, etc.)

Definitions and Examples adapted from: http://web.stanford.edu/~ouster/cgi-bin/cs142-spring12/lecture.php?topic=sql

SQL

- SQL = "Structured Query Language"
  - Non-procedural
  - Set-oriented
  - Relationally complete
  - Functionally incomplete

- Four main types of queries:
  - Insert, Delete, Select, Update

Create/drop table

- Syntax:
  - https://www.sqlite.org/lang_createtable.html
- Create a table for the students:

```
CREATE TABLE students (
id INTEGER PRIMARY KEY,
name VARCHAR(30),
birth DATE,
gpa FLOAT,
grad INTEGER
);
```

- Drop table:

```
DROP TABLE students;
```

The following failed to have auto-incremented ids: CREATE TABLE students(id INTEGER AUTO_INCREMENT, …, PRIMARY KEY (id));

Insert/Delete

- Add rows to the students table:

```
INSERT INTO students(name, birth, gpa, grad)
     VALUES ('Anderson', '1987-10-22', 3.9, 2009);
INSERT INTO students(name, birth, gpa, grad)
     VALUES ('Jones', '1990-4-16', 2.4, 2012);
INSERT INTO students(name, birth, gpa, grad)
    VALUES ('Hernandez', '1989-8-12', 3.1, 2011);
INSERT INTO students(name, birth, gpa, grad)
    VALUES ('Chen', '1990-2-4', 3.2, 2011);
```

- Delete row(s):

```
DELETE FROM students;
```

Select Queries

- 3 main elements:
  - What you want
  - Where it is found
  - How you want it filtered
- Show entire contents of a table:

```
SELECT * FROM students;
+----+-----------+------------+------+------+
| id | name      | birth      | gpa  | grad |
+----+-----------+------------+------+------+
|  1 | Anderson  | 1987-10-22 |  3.9 | 2009 |
|  2 | Jones     | 1990-04-16 |  2.4 | 2012 |
|  3 | Hernandez | 1989-08-12 |  3.1 | 2011 |
|  4 | Chen      | 1990-02-04 |  3.2 | 2011 |
+----+-----------+------------+------+------+
```

Select

- Show just a few columns from a table:

```
SELECT name, gpa FROM students;
+-----------+------+
| name      | gpa  |
+-----------+------+
| Anderson  |  3.9 |
| Jones     |  2.4 |
| Hernandez |  3.1 |
| Chen      |  3.2 |
+-----------+------+
```

- Filtering: only get a subset of the rows:

```
SELECT name, gpa   FROM students   WHERE gpa > 3.0;
+-----------+------+
| name      | gpa  |
+-----------+------+
| Anderson  |  3.9 |
| Hernandez |  3.1 |
| Chen      |  3.2 |
+-----------+------+
```

Select

- Sorting:

```
SELECT gpa, name, grad FROM students WHERE gpa > 3.0  ORDER BY gpa DESC;
+------+-----------+------+
| gpa  | name      | grad |
+------+-----------+------+
|  3.9 | Anderson  | 2009 |
|  3.2 | Chen      | 2011 |
|  3.1 | Hernandez | 2011 |
+------+-----------+------+
```

- Limiting: only get a certain number of rows:

```
SELECT name, gpa FROM students LIMIT 0,2;
+-----------+------+
| name      | gpa  |
+-----------+------+
| Anderson  |  3.9 |
| Jones     |  2.4 |
+-----------+------+
```

Update / Delete

- Update:

```
UPDATE students
    SET gpa = 2.6, grad = 2013
    WHERE id = 2;
```

- Delete:

```
DELETE FROM students
    WHERE id = 2;
```

Joins

- Join: a query that merges the contents of 2 or more tables, retrieves information from the merged results.
- Join example: many-to-one relationship
- Students have advisors; add a new table describing faculty.

```
+----+----------+-----------+
| id | name     | title     |
+----+----------+-----------+
|  1 | Fujimura | assocprof |
|  2 | Bolosky  | prof      |
+----+----------+-----------+
```

Join example: many-to-one relationship

- Add new column advisor_id to the students table. This is a ***foreign key***.

```
+----+-----------+------------+------+------+------------+
| id | name      | birth      | gpa  | grad | advisor_id |
+----+-----------+------------+------+------+------------+
|  1 | Anderson  | 1987-10-22 |  3.9 | 2009 |          2 |
|  2 | Jones     | 1990-04-16 |  2.4 | 2012 |          1 |
|  3 | Hernandez | 1989-08-12 |  3.1 | 2011 |          1 |
|  4 | Chen      | 1990-02-04 |  3.2 | 2011 |          1 |
+----+-----------+------------+------+------+------------+
```

- Perform the join query: Get the students who are advised by Fujimura

```
SELECT s.name, s.gpa FROM students s, advisors a
    WHERE a.name = 'Fujimura'   AND a.id = s.advisor_id;
```

```
+-----------+------+
| name      | gpa  |
+-----------+------+
| Jones     |  2.4 |
| Hernandez |  3.1 |
| Chen      |  3.2 |
+-----------+------+
```

```
+----+----------+-----------+
| id | name     | title     |
+----+----------+-----------+
|  1 | Fujimura | assocprof |
|  2 | Bolosky  | prof      |
+----+----------+-----------+
```

Join example: many-to-many relationship

- Courses: students take many courses, courses have many students
- Add a new table describing courses:

id	number	name	quarter
1	CS142	Web stuff	Winter 2009
2	ART101	Finger painting	Fall 2008
3	ART101	Finger painting	Winter 2009
4	PE204	Mud wrestling	Winter 2009

- Create a "join table" courses_students describing which students took which courses.

course_id	student_id
1	1
3	1
4	1
1	2
2	2
1	3
2	4
4	4

Join example: many-to-many relationship

- Find all students who took a particular course ('ART101'):

```
SELECT s.name, c.quarter
    FROM students s, courses c, courses_students cs
    WHERE c.number = 'ART101'
        AND c.id = cs.course_id
        AND cs.student_id  = s.id;
```

```
+----------+-------------+
| name     | quarter     |
+----------+-------------+
| Jones    | Fall 2008   |
| Chen     | Fall 2008   |
| Anderson | Winter 2009 |
+----------+-------------+
```

Additional Database Features

- **Indexes**: used to speed up searches
- **Concurrency**: Allow more than one program use the database at the same time.
- **Transactions**: used to group operations together to provide predictable behavior even when there are concurrent operations on the database.
- **Locks**: Used to limit concurrent access.
- **Views**: a virtual table for results of a stored query
- **Procedures**: a set of sql statements stored in the database

Exercise

- GO database
  - http://geneontology.org/page/lead-database-schema
  - http://geneontology.org/sites/default/files/public/diag-godb-er.jpg
  - http://www.berkeleybop.org/goose/
- Retrieve the names of the species that are under the genus 'Drosophila'
- Retrieve the genus and species name of the organisms whose species name has a prefix 'mel' (use LIKE function).
- Retrieve the gene symbols (gene_product.symbol) of all Drosophila melanogaster (species.genus, species.species) genes that are annotated to the 'nucleus'.
  - Join path: term.id -> graph_path.term1_id /term2_id -> association.term_id /gene_product_id -> gene_product.id
  - Join path: gene_product.species_id -> species.id

mirdb exercise

- Create a database from mirdb data.
  - http://mirdb.org/miRDB/download.html
- How many miRNAs are predicted to target XM_539064 ?
- How many predicted targets of cfa-let-7a have a prediction score of at least 80?
- List the miRNAs and the number of their targets. (Each row of the result should contain a distinct miRNA). (Use count() and GROUP BY). Show only top 10 rows of the result.