COMP 364 - Lecture 23
March 16th, 2012
Mathieu Perreault

First SQL Queries

Example database

Microarray reads
gene_id
amplitude
plate
well_num
pos_x
pos_y

Genes
gene_id
gene_name
ncbi_url

Plate
plate_id
date_run
time_run
humidity
location

GO Terms
gene_id
go_term

Database structure

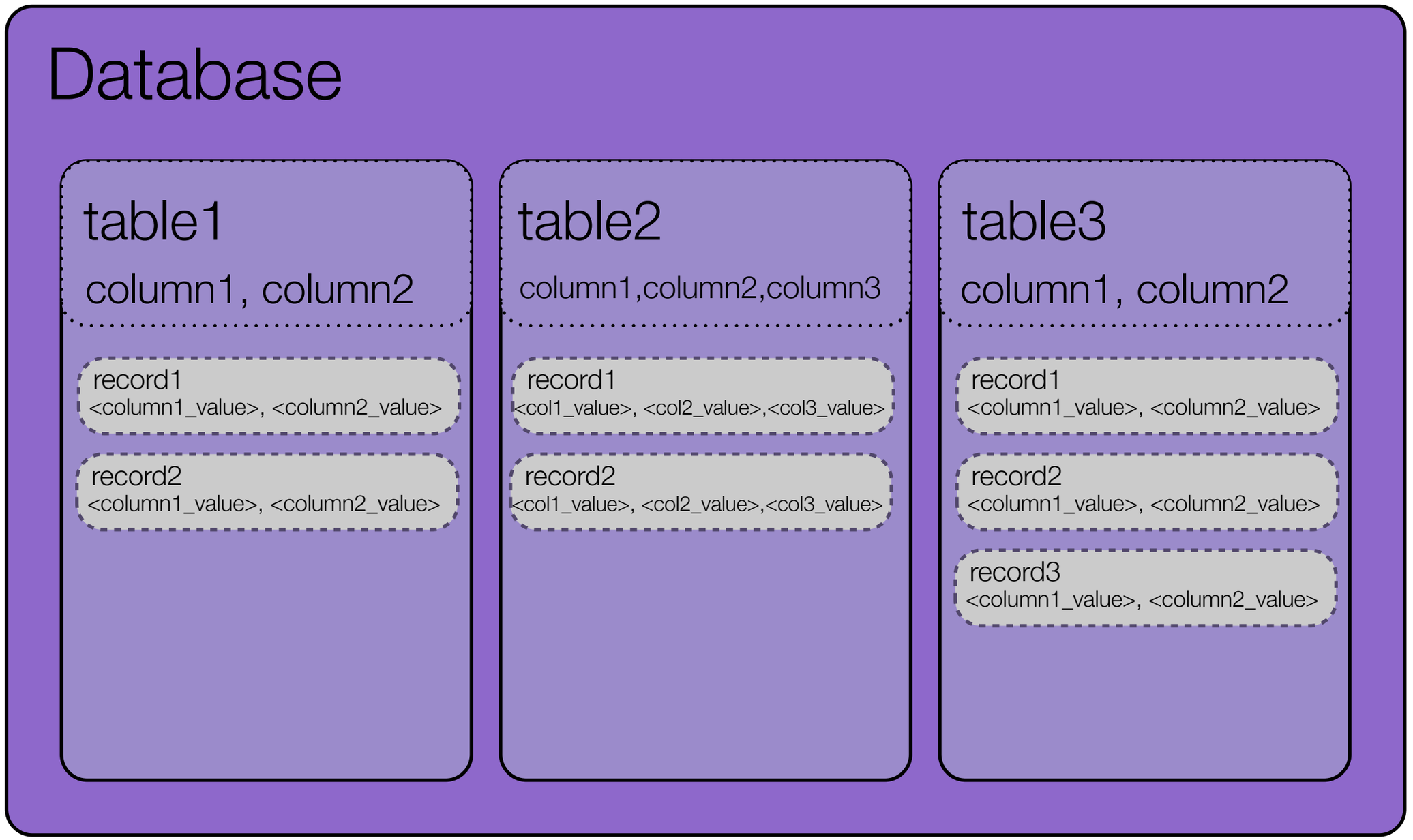
- **Database** is the main structure (only one database at a time)

- A database contains many **tables**, each with a **column structure**.
  e.g. gene_id, gene_name, gene_length for table *genes*

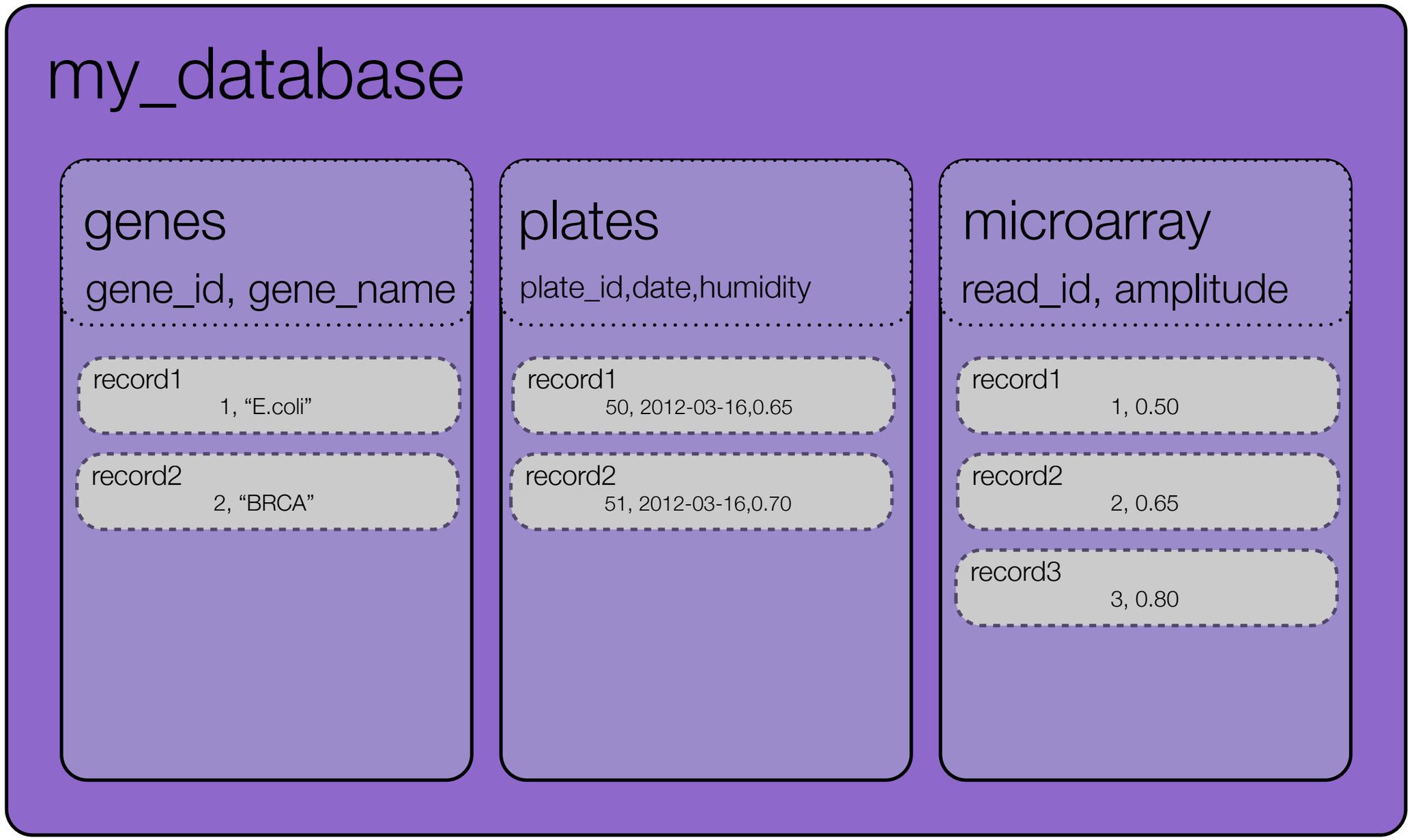
- A table can contain multiple **records**, each represented by a **row**.

- We can access the different **columns** of a record to get/set the data.

Database structure



Database structure



Creating/Opening the SQLite database

- SQLite database is just a file. Navigate to the folder and type

```
sqlite3 <database_name>
```

- If it doesn't exist, SQLite will create a new database with this name.

Some common database queries

- Opening the database
- View the tables in a database
- View the fields (columns) in a table
- View all the records (rows) in a table
- View *some* of the records according to some filtering rule
- Full SQLite Syntax reference: http://www.sqlite.org/lang.html

Microarray reads

gene_id
amplitude
plate
well_num
pos_x
pos_y

Viewing tables in a database

- Once we enter the interactive mode, we can make queries!

- First query example: viewing all the different tables that are in the database.

```
sqlite> .tables
```

Viewing columns in a table

- First let's make the output nice

```
sqlite> .header on
sqlite> .mode column
```

- Then get some information about the table

```
sqlite> PRAGMA table_info(table_name)
```

- For example:

```
sqlite> PRAGMA table_info(genes);
cid         name        type        notnull     dflt_value  pk
----------  ----------  ----------  ----------  ----------  ----------
0           gene_id     integer     0                       0
1           gene_name   varchar     0                       0
```

View all records in a table

- `SELECT [columns you want to see] FROM [table of interest];`
  - `SELECT * FROM genes;`
  - `SELECT gene_id,gene_name FROM genes;`
  - Example:

```
sqlite> SELECT amplitude FROM microarray;
amplitude
---------
0.5
0.65
0.8
```

Viewing some of the records in a table

- `SELECT * FROM tablename WHERE [conditions];`
  - `SELECT * FROM genes WHERE gene_id > 2;`
  - Example:

```
sqlite> SELECT * FROM plates WHERE plate_id = 51;
plate_id    date        humidity
----------  ----------  ----------
51          2012-03-15  0.7
```

Outputting data to a file

- You can output the results of your queries to a file, which a chosen separator.

- Example:

```
sqlite> .mode list
sqlite> .separator ","
sqlite> .output test_file_1.txt
sqlite> select * from genes;
sqlite> .exit
```

- More documentation, and many other options:

  http://www.sqlite.org/sqlite.html

Demo

Creating new content in a database

- Opening the database
- Create a table in the database
- Insert records in the database
- Update records in the database
- Delete records according to a specific condition
- Delete tables
- Full SQLite Syntax reference: http://www.sqlite.org/lang.html

Creating a table in the database

- CREATE TABLE table_name (column1 data_type1, column2 data_type2);

- Example:

```
sqlite> CREATE TABLE genes (gene_id
INTEGER PRIMARY KEY, gene_name
VARCHAR);
```

- As many columns as you want.

- Other datatypes: REAL, BLOB, NULL, TEXT

Insert records in the database

- INSERT INTO table_name VALUES (value1, value2, ...);

- Example:

```
sqlite> INSERT INTO plates VALUES(50,
“2012-03-16”, 0.60);
```

- As many columns as you want.

- You can also insert just for specific columns, and default values will be inserted for other columns:

```
INSERT INTO genes(gene_name) VALUES (“E. coli”);
```

Update some records in the database

- UPDATE table_name SET column1 = value, column2 = value2 [WHERE condition];

- Example:

```
sqlite> UPDATE genes set gene_name =
‘BRCA’ WHERE gene_id = 1;
```

- The WHERE clause is optional (but be careful!).

- Untouched columns stay the same.

Delete some (or all) records in a table

- DELETE FROM tablename [WHERE condition];

- Example:

```
sqlite> DELETE FROM genes where
gene_name= “E. coli”;
```

- WHERE clause is optional (but be careful!)

- Whole **records** are deleted at once, not just columns of a specific table (use ALTER TABLE for that).

Delete a table

- DROP TABLE [IF EXISTS] tablename;

- Example:

```
sqlite> DROP TABLE IF EXISTS genes;
```

- All the data in the table is gone, forever!

- IF EXISTS is optional, to suppress the error.