BINF 8211/6211
Design and Implementation of Bioinformatics Databases
Lecture #10

Dr. D. Andrew Carr
Dept. Bioinformatics and Genomics UNCC
Spring 2016

- What logical set operation is the Difference based on?
- How is an intersection built from a more basic logical function?
- What is the grammatical construction for using division of sets?
- If there are several attributes with the same name in two entities, how does a Natural Join determine which to compare?
- How do the Theta Join and EquiJoin differ from a Natural Join?
- What are the Aggregate Functions that I listed?
- What comparison operator would I use to compare two strings?
- What comparison operators could I use to test whether a value is between 200 and 2000?
- What expression do I use to change a name of an entity or attribute within a query (not at the database level)?

Warm-up questions

Topics

SQL as a Data definition language

Used for creation and architecture of the system.

SQL as a Data manipulation language

The easy part: SQL as a data definition language

- The schema defines the tables, views, indexes, relationships and triggers
- CREATE
  - For tables, indexes, views and trigger
- ALTER TABLE
  - Changes the structure of a table (usually to add or remove columns)
- DROP
  - Removes tables, indices, views or triggers

Create the schema

- Each SQLite database is stored in a single file.
- To create a new database file

```
$sqlite3 classdb.db
```

  - If the file exists it will be opened, if not it will be created.
- What does the schema look like of the database exists?
  - sqlite > .tables lists the table names
- Note: we are going to do an SQLite lab so I am not going to walk through all of the commands and possibilities here, just the ones relevant to the current topic.

SQLite data types with CREATE Statement

- Every entity must have a unique name and at least one column so the first step is

  sqlite > `CREATE TABLE Gene (gbid TEXT PRIMARY KEY, name TEXT, start_loc INTEGER);`

- Every attribute column must have a declared data type. SQLite uses broad categories:
  - NULL
  - INTEGER (a signed integer)
  - REAL (a floating point value)
  - TEXT (a string)
  - BLOB (a binary large object – things like images, compressed data – you won't be able to do anything but compare the label within the system.

- The .**schema** command shows the internal definition of the table.
  - sqlite > .schema Gene
  - And the system will return

    ```
    CREATE TABLE Gene (gbid TEXT PRIMARY KEY, name TEXT, start_loc INTEGER);
    ```

- Tables cannot be created if the already exist.
  - I get an error because it already exists
  - Drop the table or specify creation if exists.

  - Sqlite > `CREATE TABLE IF NOT EXISTS Gene (gbid TEXT PRIMARY KEY, name TEXT, start_loc INTEGER);`

DDL: Layout conventions

- You should specify the primary key and foreign keys as constraints

```
CREATE TABLE  genome_features
    (
        refseq_genome_id                real PRIMARY KEY,
         seqname                        text,
        source                  text,
        feature                 text,
        start_loc                       integer,
        end_loc                         integer,
        score                   real,
        strand                  integer,
        frame                   integer,
        group_name              text,
     );
```

DDL: Layout conventions

- You should specify the primary key and foreign keys as constraints

```
CREATE TABLE  genome_features
    (
        refseq_genome_id                real UNIQUE NOT NULL,
         seqname                        text,
        source                  text,
        feature                 text,
        start_loc                       integer,
        end_loc                         integer,
        score                   real,
        strand                  integer,
        frame                   integer,
        group_name              text,
     PRIMARY KEY (refseq_genome_id)
     );
```

DDL: Column Constraints

- A column definition includes
  - name
  - data type
  - constraints.

- Simple constraints
  - NOT NULL
    - no missing values
  - UNIQUE –
    - any value must be distinct from the others in that column.

  - PRIMARY KEY –
    - there can be only one column with this notation
    - (however if you want a composite PK you can specify a table constraint for PRIMARY KEY that lists them).

  - FOREIGN KEY –
    - this constraint is disabled in the default mode.
    - To turn it on you have to have compiled the library without defining SQLITE_OMIT_FOREIGN_KEY or SQLITE_OMIT_TRIGGER. Then you include a statement by database connection:

```
sqlite3> PRAGMA foreign_keys =ON;
```

  - There are foreign key ON DELETE and ON UPDATE clauses we will discuss later.

DDL: Column Constraints continued

- COLLATE –
  - specifies the name of a collating sequence/function that is used to evaluate strings (which string value is greater than another string value)
    - This can use BINARY with a memcmp() function, NOCASE folding uppercase into lowercase characters of ASCII and RTRIM which ignores trailing space characters. You can specify other functions to be called.
- DEFAULT – if no value is declared, what will be inserted (the default is null).
  - It must be a constant of the given type,
  - or a special-case keyword like CURRENT_TIME
- CHECK
  - every time a new row is added (or a row is updated) the expression registered as the CHECK constraint is evaluated

DDL: Table Constraints

- A table constraint can be
  - UNIQUE means that each row must contain a unique combination of values using attributes specified as UNIQUE.
  - CHECK – an expression for evaluating a row and comparing it to all other rows in the table can be required before data can be updated or inserted in the table.
  - WITHOUT ROWID means that the table does not use the automatically generated index as an effective primary key but only uses the primary key you have specified.

The command line

- Using the command line tool in sqlite to execute the commands from a file
  - Previously prepared .sql file

```
Sqlite > .read classdb.sql
```

```
-- SQL for classdb schema (-- comments out line)
BEGIN TRANSACTION
DROP TABLE IF EXISTS Gene;
DROP TABLE IF EXISTS Genome_Features

CREATE TABLE Gene (gbid INTEGER PRIMARY KEY, name TEXT, start_loc INTEGER);
CREATE TABLE  Genome_Features (refseq_genome_id REAL,seqname TEXT,source TEXT,feature
    TEXT,start_loc INTEGER,end_loc INTEGER,score REAL,strand INTEGER,frame
    INTEGER,group_name TEXT);

 COMMIT;
```

DDL: ATTACH

- To add your new database to the current database connection use the ATTACH DATABASE statement:
  - `sqlite > ATTACH DATABASE 'Gene.db' AS 'Gene';`
  - To make sure the system has registered it, you can list databases from the command line:
  - `sqlite > .databases`
  - Which will return

```
seq    name          file
---    --------      ----------------------------------------
0      main          /home/jweller2/programs/sqlite/classdb.db
1      Gene          /home/jweller2/programs/sqlite/Gene.db
```

DDL: INSERT and ALTER

- INSERT
  - Used to insert values
  - Slow
  - classdb. Used to specify Database
  - `sqlite > INSERT INTO classdb.Gene VALUES ('AB00145', 'SPARC2', 7600912);`

```
Sqlite > ALTER TABLE gene RENAME TO locus;
```

- ALTER
  - To add attributes
    - eg. stop position and strand

```
Sqlite> ALTER TABLE locus ADD COLUMN (stop_loc INTEGER, strand INTEGER);
```

  - SQLite does not allow:
    - rename of columnsTo check
    - Change or remove constraint

  - Use the .schema command

```
sqlite > .schema classdb.locus

CREATE TABLE locus (gbid TEXT PRIMARY KEY, name TEXT, start_loc INTEGER, stop_loc INTEGER, strand INTEGER);
```

DDL: DROP TABLE

- DROP TABLE
  - Removes table
    - Often required because FK and PK and constraints need to change.

```
sqlite > DROP TABLE IF EXISTS classdb.gene;
```

  - Any FK that is part of PK relationship
    - Prevents removal of table.

- DELETE FROM
  - removes records in a table that satisfy a condition (in the above case where a FK value exists,

Completed: SQL as a data definition language

- That is pretty much it for creating your database– read the documentation if you have questions about the command shell, syntax or limits.