SQLite Examples

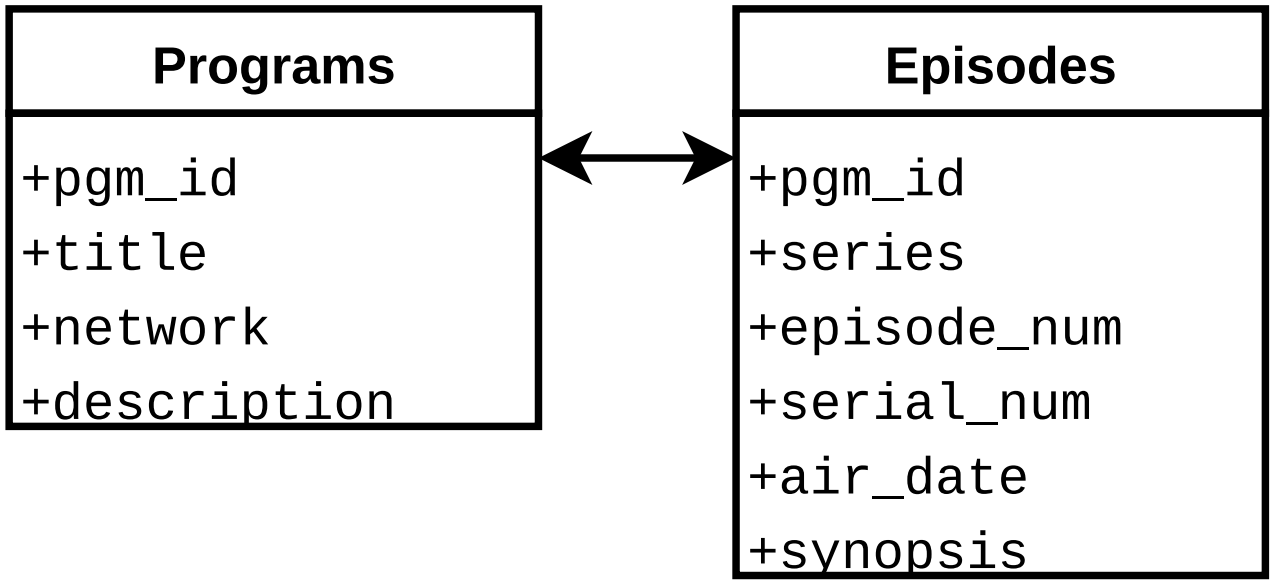
Steve Brown
Melbourne Linux Users Group
Programming SIG
http://www.mlinux.org

17 March 2009

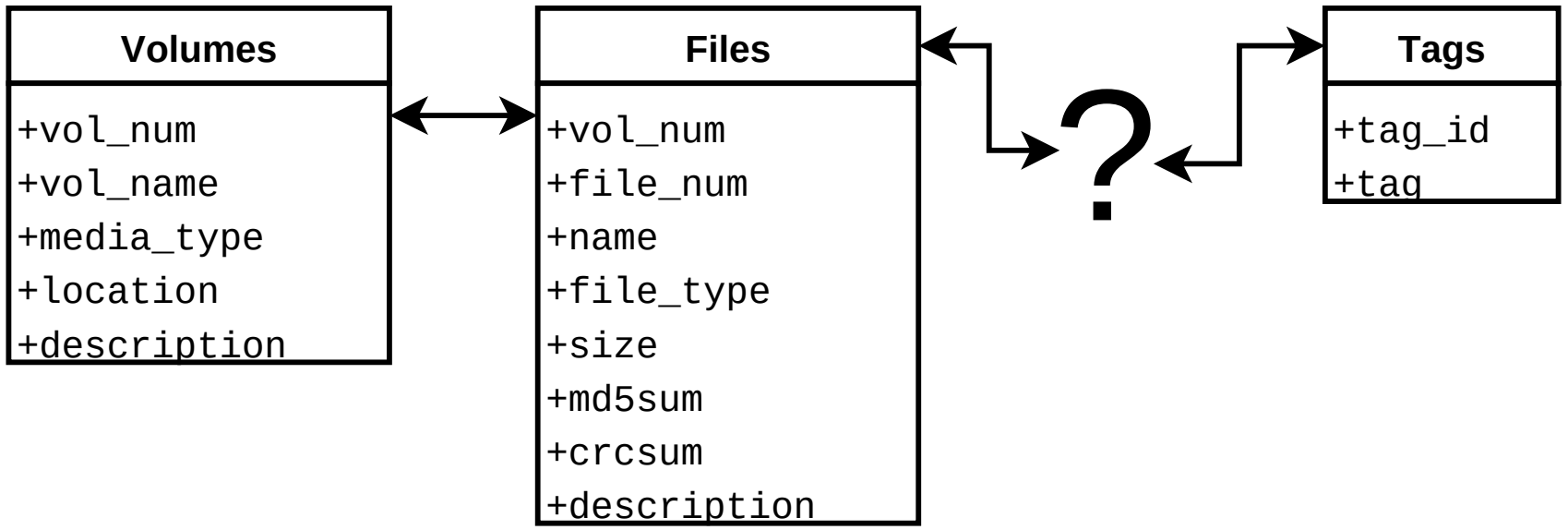
Outline

- Examples of database schema: radio/tv program index, disk archive index, recipe database, reminders
- Setting up a database: recipes
- SQLite command line interface
- SQLite Manager
- Database entry/report forms

Radio/TV Programs Database

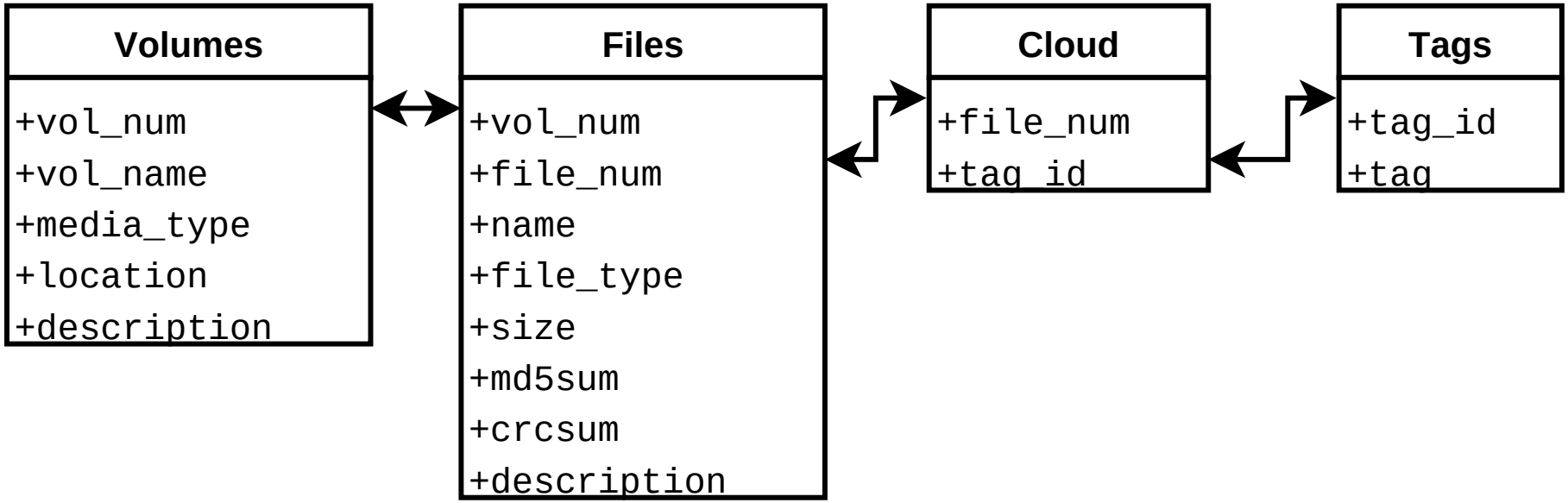


File Archive Database

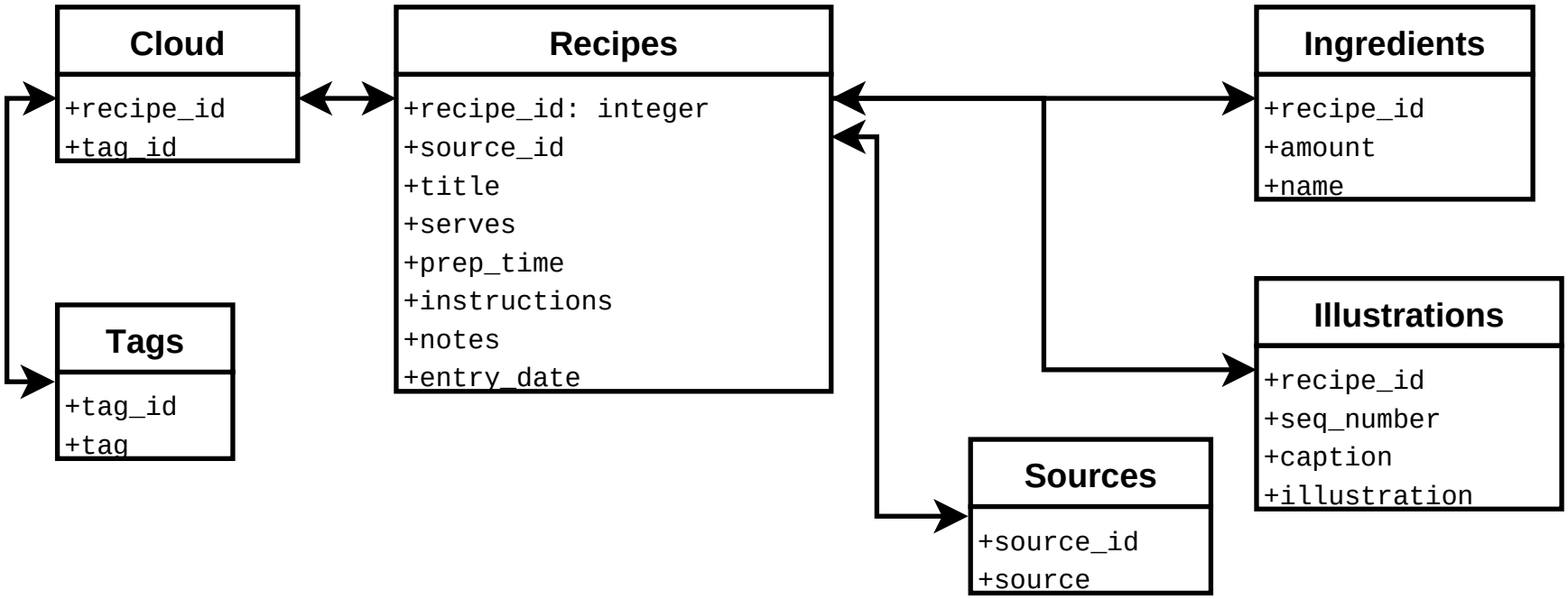


How do you link category tags with files when each file can be in many categories and each category can have many files?

File Archive Database with Tags



Recipes Database



Defining Tables in SQL

```
CREATE TABLE
    Recipes ( recipe_id INTEGER PRIMARY KEY,
       title TEXT NOT NULL COLLATE NOCASE,
       serves INTEGER DEFAULT 1,
       preparation_time REAL,
       instructions TEXT,
       source_id INTEGER REFERENCES Sources,
       notes TEXT,
       date TEXT DEFAULT CURRENT_TIMESTAMP);
CREATE TRIGGER insert_Recipes_timestamp AFTER INSERT ON Recipes
BEGIN
    UPDATE Recipes SET date=DATETIME('NOW')
        WHERE recipe_id = new.recipe_id;
END;
```

Populating Tables in SQL

```
INSERT INTO Recipes (title, instructions, notes)
    VALUES ('Beery Beef',
'Cut beef into bite size pieces.  Coat beef pieces in
flour, brown in hot oil.  Add beer and onion soup mix,
simmer until done.  Salt and pepper to taste.',
'Serve with rice or noodles.');
```

How to Build Relationships Across Tables?

- The `recipe_id` field in the `Recipes` table is populated automatically, which is important for data consistency
- But this means we don't know the `recipe_id` for a given recipe in order to `INSERT` it into the `Ingredients` table
- We have to retrieve the `recipe_id` with a query (a `SELECT`)
- There are two ways to populate `Ingredients`
  - With individual `INSERT`'s, with a `SELECT` for `recipe_id` each time
  - Build a temporary table, without `recipe_id`, and `INSERT` it all at once

Adding Ingredients (One at a Time)

```
INSERT INTO Ingredients
    SELECT recipe_id, 2, 'cups', 'sesame seeds'
    FROM Recipes WHERE Recipes.title = 'Sesame Crunch';
INSERT INTO Ingredients
    SELECT recipe_id, 0.5, 'cup', 'honey'
    FROM Recipes WHERE Recipes.title = 'Sesame Crunch';
INSERT INTO Ingredients
    SELECT recipe_id, 0.5, 'cup', 'brown sugar'
    FROM Recipes WHERE Recipes.title = 'Sesame Crunch';
INSERT INTO Ingredients
    SELECT recipe_id, 0.5, 'tsp', 'ground ginger'
    FROM Recipes WHERE Recipes.title = 'Sesame Crunch';
```

Adding Ingredients (Temporary Table)

```
CREATE TEMPORARY TABLE Single_ingredients (
   amount REAL DEFAULT 1,
   unit TEXT,
   name TEXT);
INSERT INTO Single_ingredients VALUES (2, 'lbs', 'stew beef');
INSERT INTO Single_ingredients VALUES (NULL, '', 'flour');
INSERT INTO Single_ingredients VALUES (1, 'bottle', 'beer');
INSERT INTO Single_ingredients VALUES (1, 'package',
                                       'onion soup mix');
INSERT INTO Single_ingredients VALUES (NULL, '', 'salt');
INSERT INTO Single_ingredients VALUES (NULL, '', 'pepper');
-- continued ...
```

Adding Ingredients (All at Once)

```
INSERT INTO Ingredients
    SELECT recipe_id, amount, unit, name
    FROM Recipes, Single_ingredients
    WHERE Recipes.title = 'Beery Beef';

DROP TABLE Single_ingredients;
```

Queries: Joining Tables

Build a table of recipes and their sources:

```
SELECT Recipes.title, Sources.source
    FROM Recipes,Sources
    WHERE Recipes.source_id=Sources.source_id;
```

Queries: Ingredients

Find the ingredients for a recipe:

```
SELECT Ingredients.amount, Ingredients.unit, Ingredients.name
    FROM Ingredients, Recipes
    WHERE Recipes.title = 'Sesame Crunch'
        AND Recipes.recipe_id=Ingredients.recipe_id;
```

Queries: Using Table Aliases

Find the ingredients for a recipe with less typing:

```
SELECT i.amount, i.unit, i.name
    FROM Ingredients as i, Recipes
    WHERE Recipes.title = 'Sesame Crunch'
        AND Recipes.recipe_id=i.recipe_id;
```

Queries: Following Through the Tables

Follow from a recipe title through the linking table to the corresponding tags:

```
SELECT Tags.tag FROM Tags,TagCloud,Recipes
    WHERE Recipes.title='Mulligatawny Soup'
        AND TagCloud.recipe_id=Recipes.recipe_id
        AND Tags.tag_id=TagCloud.tag_id;
```

Queries: . . . and Back the Other Way

Follow from a tag through the linking table to the corresponding recipes:

```
SELECT Recipes.title FROM Tags,TagCloud,Recipes
    WHERE Tags.tag='entree'
        AND TagCloud.recipe_id=Recipes.recipe_id
        AND Tags.tag_id=TagCloud.tag_id;
```

Queries: Using Aggregate Functions

How many recipes for each tag?:

```
SELECT count(TagCloud.recipe_id) AS tag_cnt, Tags.tag
    FROM Tags, TagCloud
    WHERE Tags.tag_id=TagCloud.tag_id
        GROUP BY Tags.tag
        ORDER BY tag_cnt DESC;
```

Using SQLite Manager with Firefox

- Demonstration: browse to
  `https://addons.mozilla.org/en-US/firefox/addon/5817`
  to install the SQLite Manager Firefox add-on

SQLite & Java

- There are at least two options to use SQLite with Java:
- JNI (Java native interface)
  - This is a wrapper around the SQLite C++ library
  - It preserves the SQLite features, but is not readily portable to other relational database back-ends
- JDBC
  - This is a Java standard for relational database interfaces
  - It should be fairly portable across different relational database back-ends

SQLite & Java I: JNI Wrapper

- This is a homework assignment. . .

SQLite & Java II: JDBC Driver

- Java provides a standard interface to relational databases called *the JDBC API*
- The JDBC API is contained in two packages:
  – Most of it is in *java.sql*, a few additional utilities are in *javax.sql*
- To use this with any given database, you need a JDBC driver for that type of database (e.g. SQLite)

JDBC Example

Grab *Recipe.java* and *sqlitejdbc-v54.jar* and compile and run the application:

```
javac Recipe.java
java -cp .:sqlitejdbc-v054.jar Recipe beery sesame
```

SQLite & Python

- If you have Python 2.5 or later, SQLite bindings (conforming to the Python DB API spec) are in the library, in the `sqlite3` module
  - For earlier versions, you need to install the `pysqlite` or `pysqlite2` module
- To use a database, you need to create a *connection* object to open the database
- Then you use a *cursor* object to execute SQL statements and iterate over result sets
- Note that the Python DB API and SQLite differ slightly in the convention for how transactions are structured—follow the Python documentation to see how

Python Example

Grab *Recipes.py* and run the application:

```
chmod u+x Recipes.py
./Recipes.py beery sesame
```

References

- SQLite: `http://www.sqlite.org/`
- SQLite Manager:
  `https://addons.mozilla.org/en-US/firefox/addon/5817`
- SQLite JNI wrapper: `http://www.ch-werner.de/javasqlite/index.html`
- SQLiteJDBC: `http://www.zentus.com/sqlitejdbc/`
- pysqlite2: `http://www.pysqlite.org/`
  or `http://www.python.org/doc/`