**Python for Everyone: Notes**

**Chapter 15: Relational databases and SQLite**

* Random Access
  + When you can randomly access data
  + How can you layout data to be most efficient
  + Sorting might not be the best idea
* Relational database
  + Relational databases model data by storing rows and columns in tables. The power of the relational databases lies in its ability to efficiently retrieve data from those tables and in particular where there are multiple tables and the relationships between those tables involved in the query
* Terminology
  + Database- contains many tables
  + Relation (or table) – contains tuples and attributes
  + Tuple (or row) – a set of fields that generally represents an “object” like a person or a music track
  + Attribute (also column or field) – one of the possibly many elements of data corresponding to the object represented by the row
* A relation is defined as a set of tuples that have the same attributes. A tuple usually represents an object and information about that object. Objects are typically physical objects or concepts. A relation is usually described as a table, which is organized into rows and columns. All the data referenced by an attribute are in the same domain and conform to the same constraints
* SQL
  + Structured query language is the language we use to issue commands to the database
    - Create data (insert)
    - Retreieve data
    - Update data
    - Delete data
* Web applications w/ databases
  + Application developer – builds the logic for the application, the look and feel of the application – monitors the application for problems
  + Database administrator – monitors and adjust the database as the program runs in production
  + Often both people participate in the building of the data model
* Database administrator
  + A database administrator (dba) is a person responsible for the design, implementation, maintence, and repair of an organization’s database. The role includes the development and design of database strategies, monitoring and improving database performance and capacity, and planning for future expansion requirements. They may also plan, coordinate, and implement security measure to safeguard the database
* Database model
  + A database model or database schema is the structure or format of a database, described in a formal language supported by the database management system. In other words, “a database model” is the application of a data model when used in conjuction with a database management system.
* Common database system
  + Three major database management systems in wide use
    - Oracle – large, commercial enterprise-scale, very very tweakable
    - Mysql – simpler but very fast and scalable – commercial open source
    - Sqlserver – very nice – from Microsoft (also access)
  + Many other smaller progjects, free and open source
* SQLite browser
  + Sqlite is a very popular database- it is free and fast and small
  + Sqlite browser allows us to directly manipulate sqlite files
  + Sqlite is embedded in python and a numer of languages
* Sql: insert
  + The insert statement inserts a row into a table
  + INSERT INTO Users (name, email) VALUES (‘Kristin’, ‘kf@umich.edu’)
* SQL: delete
  + Deletes a row in a table based on selection criteria
  + DELETE FROM Users WHERE email=’ted@umich.edu’
* SQL: update
  + Allows the updating of a field with a where clause
  + UPDATE Users SET name=’Charles’ WHERE email=’csev@umich.edu’
* Retrieving records: Select
  + The select statement retrieves a group of records – you can either retrieve all the records of a subset of the records with a WHERE clause
  + SELECT \* FROM Users
  + SELECT \* FROM Users WHERE email=’csev@umich.edu’
* Sorting with order by
  + You can add an ordered by clause to select statements to get the results sorted in ascending or descending order
  + SELECT \* FROM Users ORDER BY email
  + SELECT \* FROM Users ORDERED BY name DESC
* SQL Summary
  + INSERT INTO User (name, email) VALUES (‘Kristin’, ‘kf@umich.edu’)
  + DELETE FROM Users WHERE email=’ted@umich.edu’
  + UPDATE Users SET name =’Charles’ WHERE email=’csev@umich.edu’
  + SELECT \* FROM Users
  + SELECT \* FROM Users WHERE email=’csev@umich.edu’
  + SELECT \* FROM User ORDER BY email
* This is not too exciting so far
  + Tables pretty much look like big fast programmable spreadsheets with rows, columns, and commands
  + The power comes when we have more than one table and we can exploit the relationships between the tables
* Database design
  + Database design is an art form of its own with particular skills and experience
  + Our goal is to avoid the really bad mistakes and design clean and easily understood databases
  + Others may performance tune things later
* Building a data model
  + Drawing a picture of the data objects for our application and then figuring out how to represent the objects and their relationships
  + Basic rule: don’t put the same string data in twice – use a relationship instead
  + When there is one thing in the real world there should be one copy of that thing in the database
* For each piece of info
  + Is the column an object of an attribute of another object
  + Once we define objects, we need to define the relationship between objects
* Representing relationships in a database
* Database normalization (3nf)
  + There is tons of database theory – way too much to understand without excessive predicate calculus
  + Do not replicate data – reference data – point at data
  + Use integers for keys and for references
  + Add a special “key” column to each table which we will make references to. By convention, many programmers call this column “id”
* We want to keep track of which band is the “creator” of each music track
  + What album does this song “belong to”
  + Which album is this song related to
* Integer reference pattern
  + We use integer to reference rows in another table
* Three kinds of keys
  + Primary key – generally an integer auto – increment field
  + Logical key – what the outside world uses for lookup
  + Foreign key – generally an integer key pointing to a row in another table
* Key rules
  + Best practices
    - Never use your logical key as the primary key
    - Logical keys can and do change, albeit slowly
    - Relationships that are based on matching strings fields are less efficient than integers
* Foreign keys
  + A foreign key is when a table has a column that contains a key which points to the primary key of another table
  + When all primary keys are integers, then all foreign keys are integers – this is good -very good
* Using join across tables
* Relational power
  + By removing the replicated data and replacing it with references to a single copy of each bit of data we build a “web” of information that the relational database can read through very quickly – even for very large amounts of data
  + Often when you want some data it comes from a number of tables linked by these foreign keys
* The JOIN operation
  + The join operation links across several tables as part of a select operation
  + You must tell the JOIN how to use the keys that make the connection between the tables using an ON clause
* Select Album.title, Artist.name from Album join Artist on Album.artist\_id=Artist.id
  + What we want to see
  + The tables that hold the data
  + How the tables are linked
* SELECT Track.title, Track.genre\_id, Genre.id, Genre.name FROM Track JOIN Genre
  + Joining two tables without an ON clause gives all possible combination of rows
* Many – to – many relationships
  + Sometimes we need to model a relationship that is many-to-many
  + We need to add a “connection” table with two foreign keys
  + There is usually no separate primary key
* Complexity enables speed
  + Complexity makes speed possible and allows you to get very fast results as the data size grows
  + By normalizing the data and linking it with integer keys, the overall amount of data which the relational database must scan is far lower than if the data were simply flattened out
  + It must seem like a tradeoff – spend some time designing your database so it continues to be fast when your application is a success
* Additional SQL topics
  + Indexes improve access performance for things like string fields
  + Constraints on data – Cannot be null, etc
  + Transactions – allows sql operations to be grouped and done as a unit