BS803 Introduction to SQL

Fall 2022

Brian Gregor
bgregor@bu.edu

Research Computing Services
rcs.bu.edu

BOSTON UNIVERSITY

Outline

- Set up your computer for using SQL
- Relational Database concepts
- SAS & SQL
- Basic SQL queries
- Normalization
- SQL Joins
- Multi-step queries

Setup for SQL

On the SCC a similar program is available via a module:
    sqlitebrowser/3.10.1

- Visit DB Browser.
- Download the Mac or Windows installer & install it.
- Get the database:
  - http://rcs.bu.edu/examples/db/tutorials/bs803/data.zip
  - Unpack that ZIP file somewhere on your computer
    - Windows users: right-click the file, choose *Extract All*

  - The data is the Zillow Home Value Index for US cities from 2000-2022.

Relational Databases

- From the Oxford English Dictionary:

“A structured set of data held in computer storage and typically accessed or manipulated by means of **specialized software**.”

- This is distinct from an application (i.e. your SAS, R, or Python program) reading, storing, and manipulating data from a stored dataset.

Parts of "the database"

Stored Data
(on disk, in memory)

database

Software managing
the data

Database management
software (DBMS)

Database system

Computer running
the DBMS

Database server

The common term "the database" usually
refers to the combo of a DBMS + database.

Stored Data

- Data in a database can be stored in any number of ways with varying degrees of sophistication.
- Usually, it's in a format specific to the DBMS
  (our focus for today)
- But it could also be:
  - A single (giant) text file
  - A collection of Excel spreadsheets
  - Directories and subdirectories of image files or other data files
  - Etc.

We could have a database that provides sorting and ordering for a *dataset* that consists of these types of data.

E.g. Excel Power Query

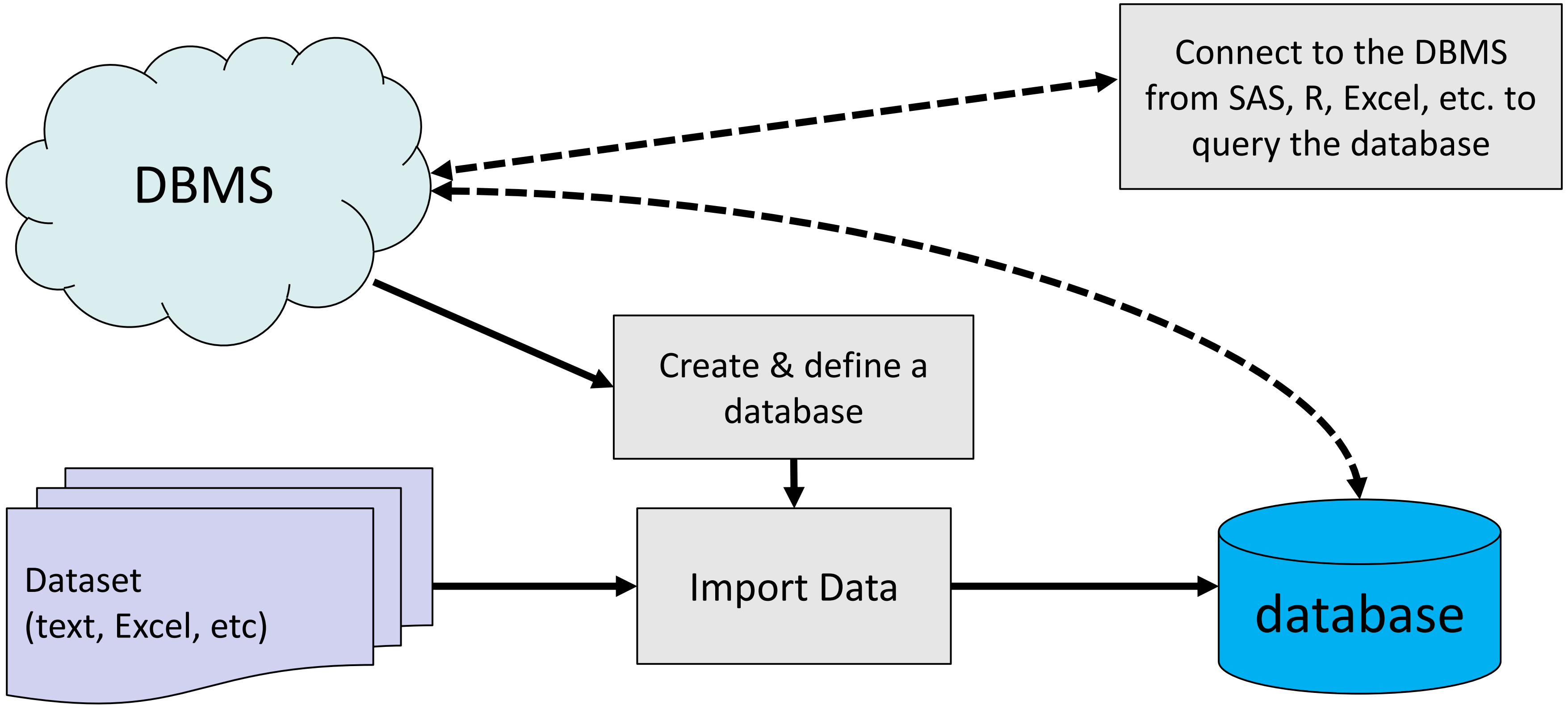
BOSTON UNIVERSITY

Database Management Software

- The DBMS provides functionality to manage, retrieve, and store data in the database.
- Most provide multi-user access to a database over networks.
- A single DBMS program can typically handle **multiple** databases
  - And serve them to different groups of users or applications
  - Example: a single DBMS hosts databases for engineering, sales, and customer info

Database Management Software

- The storage, creation, and management of the database is handled by the DBMS


DBMS
Connect to the DBMS from SAS, R, Excel, etc. to query the database
Create & define a database
Dataset (text, Excel, etc)
Import Data
database

Database Servers

- A single application:
  - Use a DBMS via a software library to handle data storage & retrieval from a database.
- Separate DBMS for an application:
  - A computer runs a DBMS to provide data for a web server on the computer.
- Dedicated hardware:
  - Large scale datasets or large numbers of users need a computer (or cluster) dedicated to the DBMS while the applications run elsewhere.

BOSTON UNIVERSITY

When to use a database…

- You have more data than your application can conveniently access
  - E.g. the dataset is larger than the amount of RAM on your computer.
- You need to search through your data
  - Especially when the searches are complex
  - Or involve internal relationships between parts of your dataset
- You want your data storage to be reliable and fault-tolerant.
  - If your DBMS is properly ACID compliant it is ***very hard*** to corrupt its data storage or store invalid or incomplete data sets.

BOSTON UNIVERSITY

When to use a database…

- You have more data than your application can conveniently access
- The data is well-structured
  - E.g. timestamped, clear types – users, items, images, etc., internal relationships between datasets
- Your application needs to scale up to handle large quantities of data
- You need to search through your data
  - Especially when the searches are complex
  - Or involve internal relationships
- You want your data storage to be reliable and fault-tolerant.
  - If your DBMS is properly ACID compliant it is ***very hard*** to corrupt its data storage or store invalid or incomplete data sets.

BOSTON UNIVERSITY

Structured Query Language (SQL)

- A programming language that is executed by a RDBMS to perform queries
  - SQL statements can read, insert, or delete data from the database, create or delete tables, manage user accounts for the database, store queries to run as functions, and more.
- The queries define where to search (which tables, etc.) and what to return
- Queries do not define **how** to do the search – the algorithms depend on the RDBMS implementation
- Query results can be data from the database or derived values created during the query.
- SQL as a language standard is *mostly* adhered to by DBMS vendors.
  - ~90% compatible across different SQL systems.

BOSTON UNIVERSITY

Popular SQL RDBMS systems

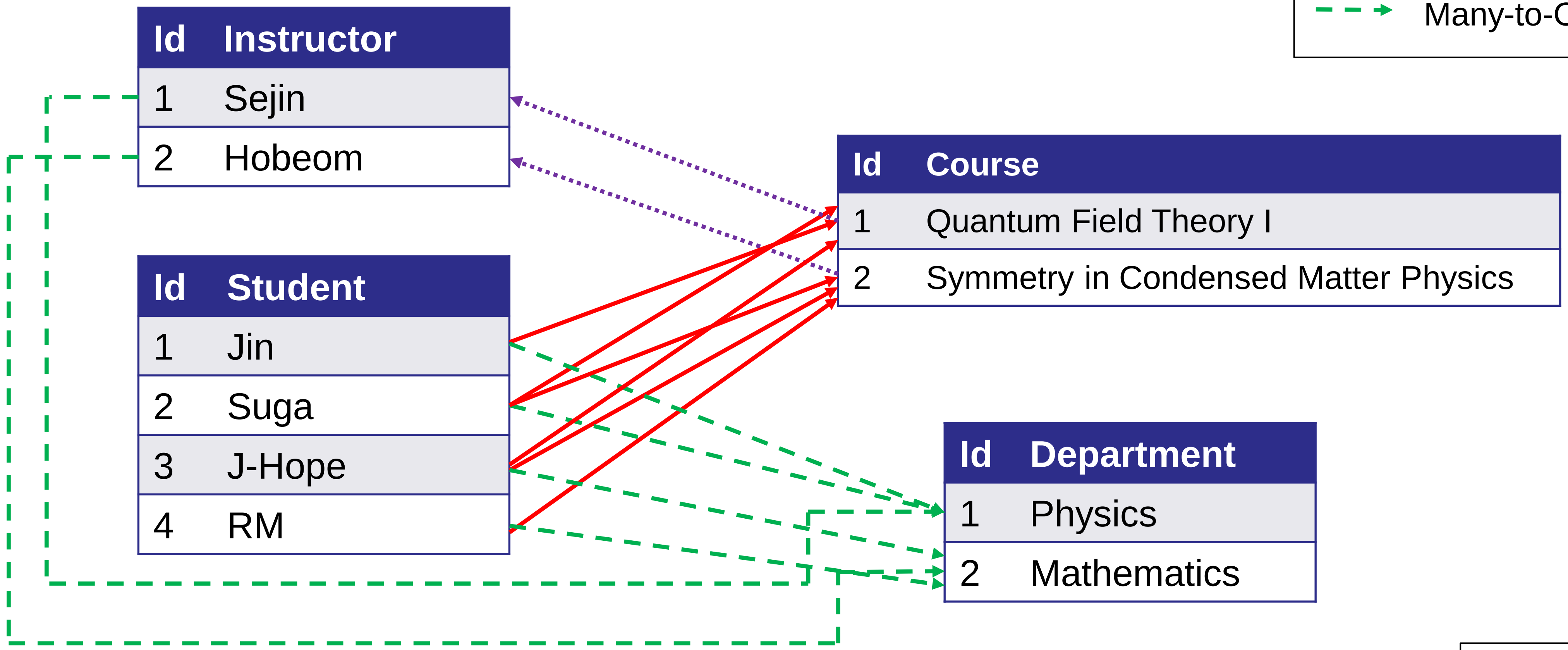
Name	Max database size	Multi-User	Open Source?	Server/Client?	OS Platforms*
SQLite	140 TB	read only	yes	No	All of them! (incl. iOS, Android)
PostGreSQL	151 ExaBytes ($151x10^6$ TB)	read / write	yes	yes	Windows, Linux, Mac OSX
MySQL	Default is 256 TB, max 65,536 TB	read / write	yes	yes	Windows, Linux, Mac OSX
Microsoft SQL Server	(free) 10 GB ($) 524,272 TB	read / write	no	yes	Windows, Linux
Oracle DB	(free)11 GB to “how big is your wallet?”	read / write	no	yes	Windows, Linux, other Unix

BOSTON UNIVERSITY

* To run the DBMS. Client access can be from any platform.

A Relational Database Example

- ⋯▸ One-to-One
- ⟶ Many-to-Many
- - - ▸ Many-to-One



Id	Instructor
1	Sejin
2	Hobeom

Id	Student
1	Jin
2	Suga
3	J-Hope
4	RM

Id	Course
1	Quantum Field Theory I
2	Symmetry in Condensed Matter Physics

Id	Department
1	Physics
2	Mathematics

Tables and their relationships are defined in a *database schema*

SAS & SQL

- SAS has SQL built in. You can use it to query SAS datasets.
  - It’s a subset of the SQL language, limited to *SELECT* statements which perform queries.

```
proc sql;
    select BookingDate, ReleaseDate,
    ReleaseCode from SASclass.Bookings;
quit;
```

  - See their docs for info on how to use this.

- SAS can also be connected to external SQL databases using the SAS/ACCESS add-on.

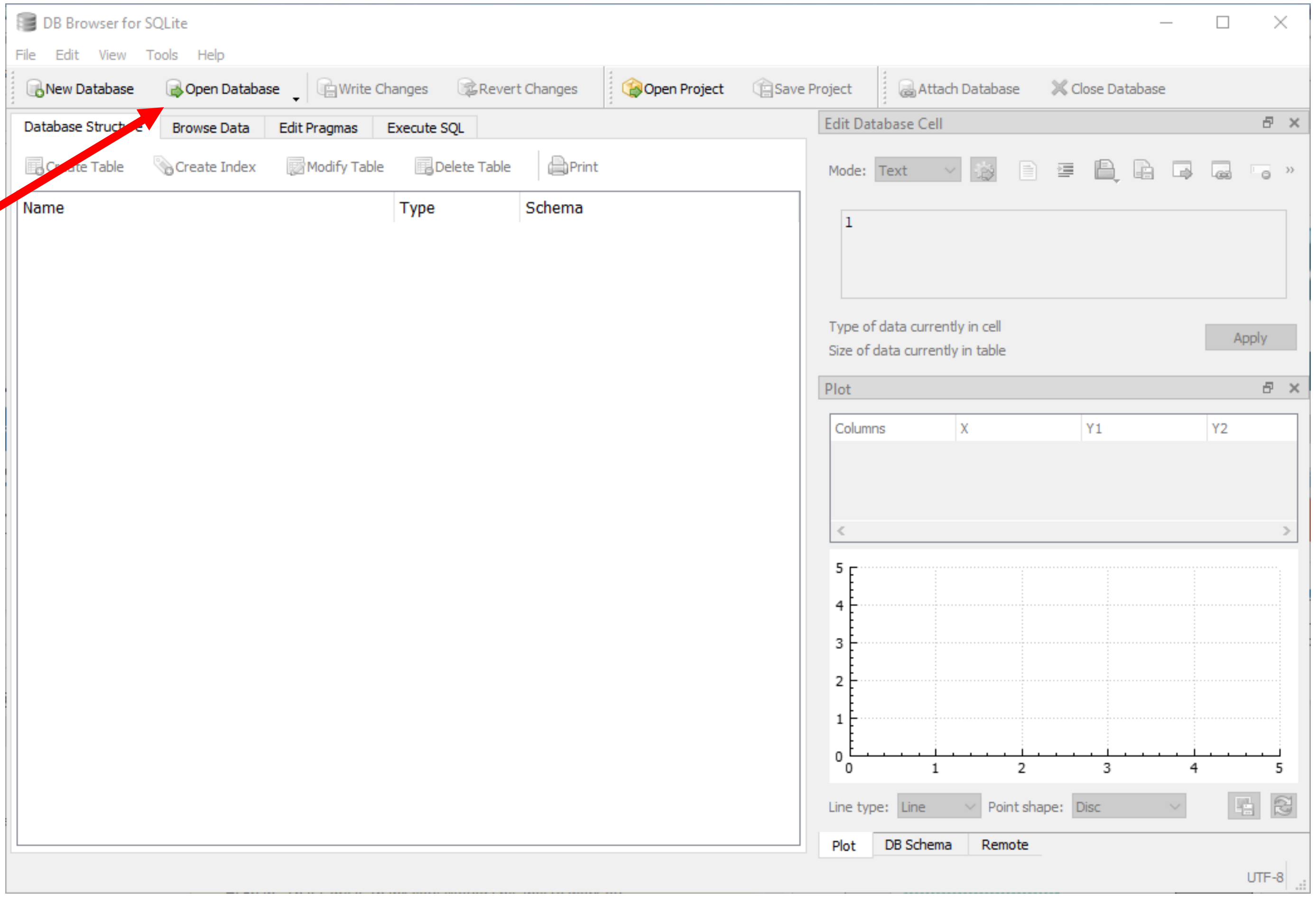
SQLite

What Is SQLite?
SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day.

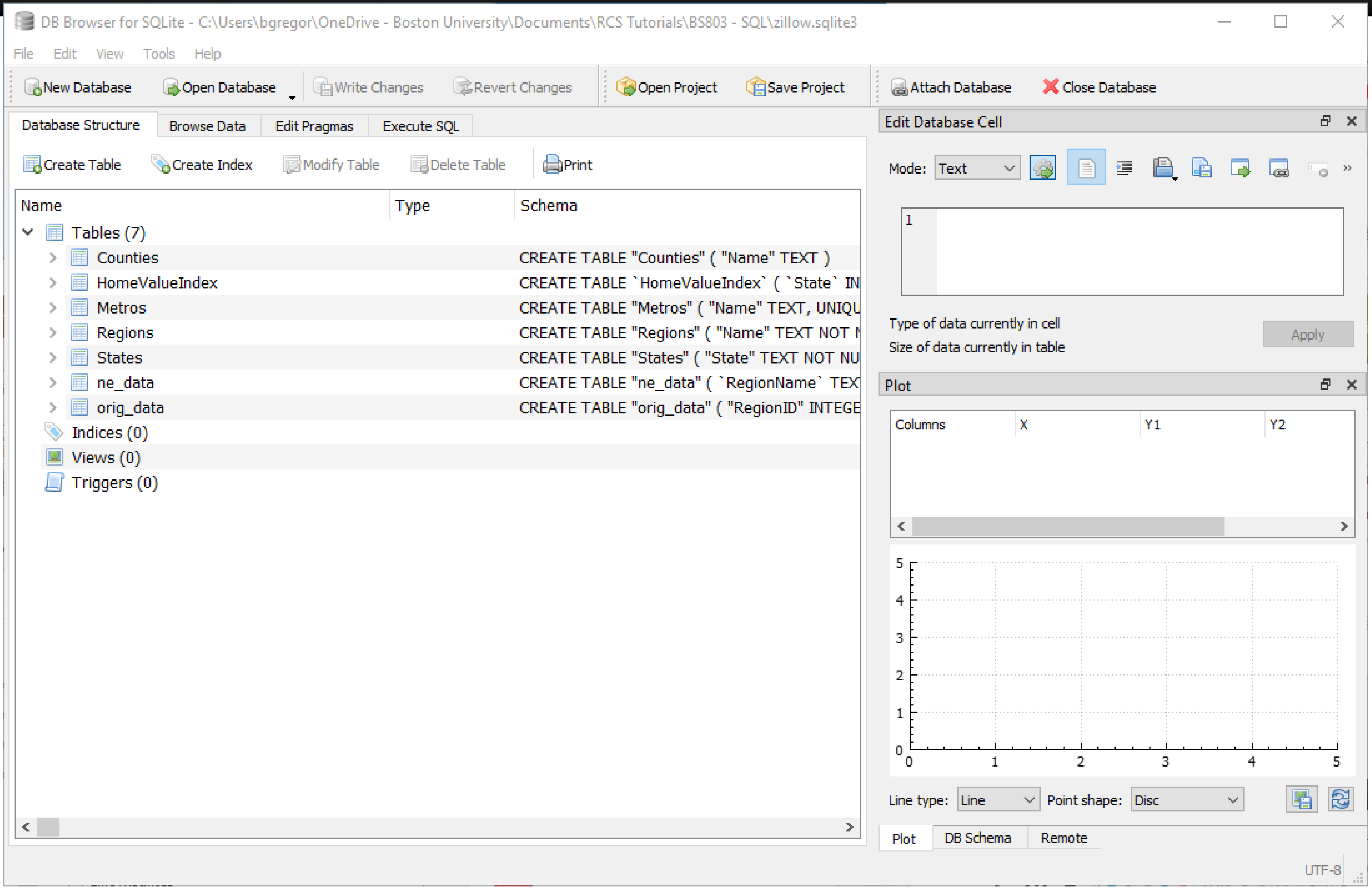
- There are some key differences with SQLite vs. other DBMS systems.
- Single user (single application), no concept of users, small number of datatypes, dynamic typing.

Open DB Browser

- Click “Open Database”
- Choose the “zillow.sqlite3” file.



BOSTON UNIVERSITY


DB Browser for SQLite - C:\Users\bgregor\OneDrive - Boston University\Documents\RCS Tutorials\BS803 - SQL\zillow.sqlite3

Name	Type	Schema
Tables (7)		
Counties		CREATE TABLE "Counties" ("Name" TEXT)
HomeValueIndex		CREATE TABLE `HomeValueIndex` (`State` IN
Metros		CREATE TABLE "Metros" ("Name" TEXT, UNIQU
Regions		CREATE TABLE "Regions" ("Name" TEXT NOT N
States		CREATE TABLE "States" ("State" TEXT NOT NU
ne_data		CREATE TABLE "ne_data" (`RegionName` TEX
orig_data		CREATE TABLE "orig_data" ("RegionID" INTEGE
Indices (0)		
Views (0)		
Triggers (0)		

BOSTON UNIVERSITY

- DB Browser should now look more or less like this.

The zillow.sqlite3 database

- Tables:
  - orig_data
    - an import of the CSV file
  - ne_data
    - A modification of orig_data with New England data
  - Counties, Metros, Regions, States, HomeValueIndex
    - A radical reorganization of the orig_data table.

- orig_data: contains columns that specify place (StateName, RegionName, Metro) and columns named for dates that contain house price estimates (“Home Value Index”).

SELECT: a read-only query

SQL syntax is indifferent to capitalization. The convention is to capitalize SQL keywords.

Only 1 table name

SELECT columns FROM table_name WHERE condition … ;

Column names from the table
* Means all columns

Boolean conditions on columns that filter the results.

Other keywords, stay tuned.

BOSTON UNIVERSITY

SQL Comments

- Comment your queries:

```
-- A single line comment

/*  This is
    a multi-line
    comment */
```

Enter & Run

- `SELECT State, RegionName FROM ne_data ;`
  - Returns 1428 rows with 2 columns.
- `SELECT State, RegionName FROM ne_data LIMIT 10;`
  - Stop after 10 rows. This is handy for developing/debugging queries.
- `SELECT State, RegionName FROM ne_data WHERE State= 'MA' ;`
  - The WHERE condition filters the result to just Massachusetts.

-
  ```
  SELECT State, RegionName FROM ne_data
              WHERE Jan2010 > 500000 ;
  ```
  - Here filter to home values > $500k.

BOSTON UNIVERSITY

Poor Naming scheme in *orig_data*

- Most column names in **orig_data** start with numbers. This confuses SQL.
- Try adding single quotes
  - SQL usually uses single quotes for strings

-
  ```
  SELECT State,RegionName FROM orig_data WHERE
  '2007-06-30' > 1000000 ;
  ```

- That's comparing a string to an integer. Try telling SQL that the string really is part of the table:

-
  ```
  SELECT State,RegionName FROM orig_data WHERE
      orig_data.'2007-06-30' > 1000000 ;
  ```

Table_name.'column_name'
will clear up ambiguity.

BOSTON UNIVERSITY

WHERE operators

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal (Sometimes != is allowed)
BETWEEN	Between a certain range
LIKE	Pattern search
IN	Multiple options
AND, NOT, OR	Apply multiple conditions

BOSTON UNIVERSITY

Add more clauses with Boolean operators

-
  ```
  SELECT State,RegionName, Jan2005, Jan2010
       FROM ne_data
       WHERE Jan2010 > 500000
       AND (State='ME' OR State='NH’);
  ```

-
  ```
  SELECT State,RegionName, Jan2005, Jan2010
       FROM ne_data
       WHERE Jan2010 > 500000
       AND  State IN ('ME','NH’);
  ```

BOSTON UNIVERSITY

Aggregations

- We can aggregate results by applying functions to columns:
- `SELECT State,RegionName, MAX(Jan2010) FROM ne_data ;`
- Rename with the AS keyword
- `SELECT State,RegionName, MAX(Jan2010) AS MaxValue FROM ne_data ;`

Or by apply GROUP BY

- `SELECT Count(*) FROM ne_data WHERE State='MA'`
  - Returns 364, meaning there are 364 rows where the state is Massachusetts.
- Now do this for all states:

- `SELECT State,COUNT(*)  FROM ne_data GROUP BY State;`

- Try to get the state with the largest number of entries in the table…

BOSTON UNIVERSITY

- Treat the SELECT like it’s a table and do a SELECT on it…you can’t nest an aggregate function inside another like MAX(COUNT(*))

-
  ```
  SELECT State, MAX(C) FROM
  (SELECT State,COUNT(*) AS C FROM ne_data
  GROUP BY State)
  ```

BOSTON UNIVERSITY

SQLite functions

- Core functions – string manipulation, basic math, etc.
- Aggregate functions – min, max, sum etc.
- Math functions – rounding, trig functions, log, etc.
- Date/Time – functions dealing with times stored as ISO-8601, Julian numbers, or seconds since the Unix epoch.
- Window functions – input values are taken from some subset of rows as returned by a SELECT.

Average Home Values

- Try: What are the average home values by county (CountyName) for 2005 and 2010?

- Sort your query in descending order by adding an **ORDER BY**:

- `SELECT State,COUNT(*) AS C  FROM ne_data GROUP BY State` **`ORDER BY`** `C` **`DESC;`**

```
ORDER BY col ASC
ORDER BY col DESC
```

Choose a direction, ascending or descending.

Complex queries

- Database queries can be ***very*** long.
- You can create your own tables for convenience with the CREATE TEMPORARY TABLE command. Temporary tables are removed at the end of the query.
  - Create a few temp tables, combine them, select from them to return a final result.
- SELECT results can sometimes be assigned to variables.
  - Depends on the DBMS. SQL Server allows this, for example.
- It's like writing a SAS, R, or Python program…!

BOSTON UNIVERSITY

UPDATE

- To update a value (or row, or set of rows) in a table use the UPDATE command:

- UPDATE table_name

  SET column1 = value1, column2 = value2, ...

  WHERE condition;

- But wait – what happens if an error occurs?
  - Illegal value entered (string in place of an int, required value not provided, power plug yanked etc)

Transactions

- SQL databases support transactions, which are all-or-nothing changes to the database
  - Every part of the transaction succeeds or the database is unchanged.

- Many times these are enabled automatically by libraries, so queries from R, Python, or SAS are handled as transactions. DB Browser uses these in its *Execute SQL* editor.

```
BEGIN; --or BEGIN TRANSACTION ;
UPDATE ne_data SET Jan2005=10*Jan2005
               WHERE RegionName='Boston' ;

-- ROLLBACK – cancel this transaction

COMMIT; -- OR END
```

Normalization

- This is the name of the process that reduces redundancy and improves data integrity in a database.
- Let’s look at the *orig_data* table.
- We want to have fixed definitions (like state or county names) to be specified exactly once in the database.
- The date columns are **very** hard to use, let’s a make a new table that stores all the date-based home values.

A new schema

- *Counties* – stores the county names
- *Metros* – stores metro (city) names
- *Regions* – store region names
- *States* – store state names and abbreviations
- *HomeValueIndex* – Store **all** of the home values for all places and dates.

- Let’s look at the set of SQL files (and 1 Python file) that implements this change.
  - Demonstrates the CREATE TABLE and INSERT commands to define tables and add rows to them.

BOSTON UNIVERSITY

Importing Data

- A common scenario: you have some dataset (CSV, etc.) that needs to be added to a normalized database. This process might look like this:
  - Create a temporary table that mirrors the raw data
  - Insert the raw data into the temporary table.
  - Use SELECT, INSERT, and UPDATE and maybe one or more other temporary tables to re-arrange the raw data into a format that is compatible with the database scheme.
  - Do an INSERT or UPDATE into the database tables of the raw data.
  - Query ends, temporary tables are automatically deleted.

BOSTON
UNIVERSITY

Relationships

- The **FOREIGN KEY** is used to create relationships.

- 1-to-1 or 1-to-many:
  - Table A has a FOREIGN KEY that references a **rowid** or other primary key to table B.

- “Real” 1-to-1:
  - Table A has a FOREIGN KEY that references a **rowid** or other primary key to table B *and* vice-versa.
- Many-to-many:
  - Create a third table C that contains 2 columns: a FOREIGN KEY referencing table A and a FOREIGN KEY referencing table B. Table C is sometimes called a “jump” table.
  - Multiple rows in table C create multiple relationships between tables A and B.

BOSTON UNIVERSITY

Joins

- In order to build queries we now need to attach our tables together.

```
SELECT HomeValueIndex.State as StateID, States.State
FROM HomeValueIndex
INNER JOIN States ON HomeValueIndex.State=States.rowid
LIMIT 10 ;
```

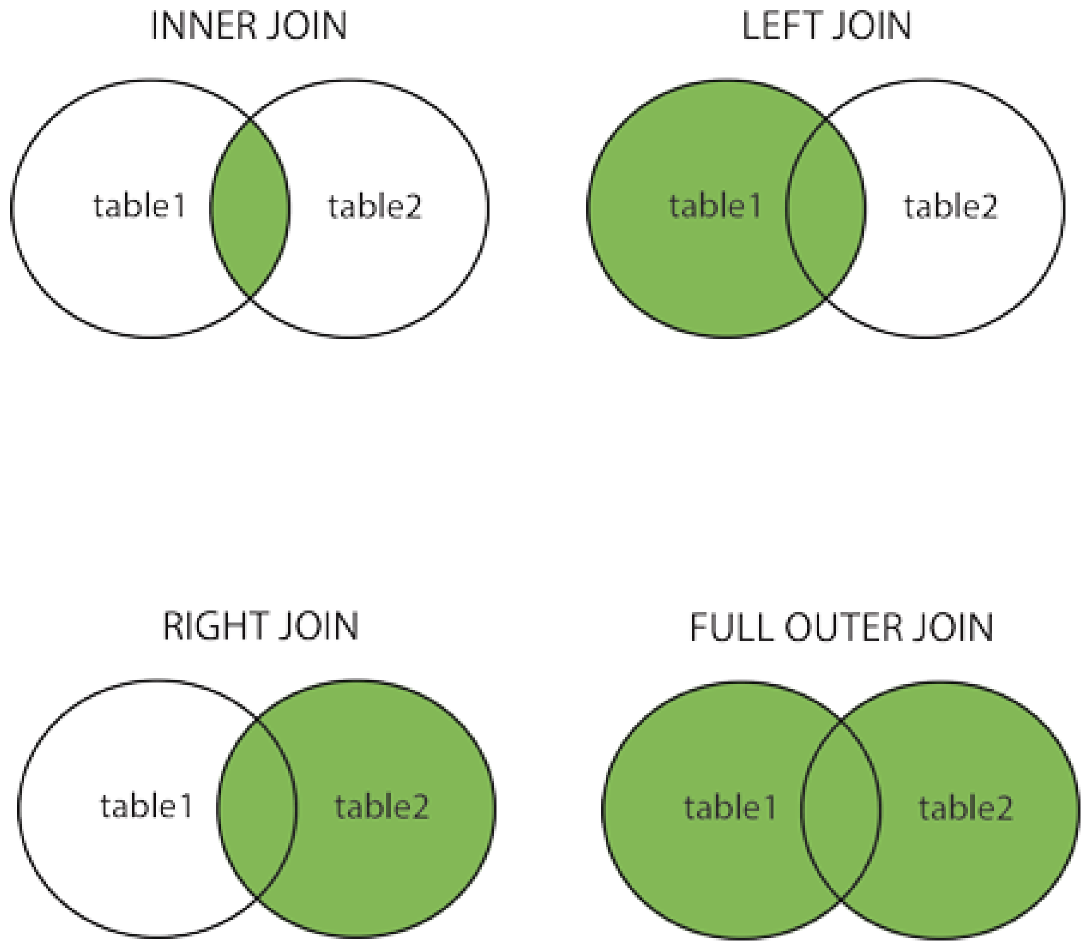
- You can rename tables for convenience

```
SELECT H.State as StateID, S.State
FROM HomeValueIndex H
INNER JOIN States S ON H.State=S.rowid
LIMIT 10 ;
```

BOSTON UNIVERSITY

Join Types

- INNER JOIN (or just JOIN): Join condition is satisfied in both tables.
- LEFT JOIN: All matches from the left table, matched records from the right or NULL values.
- RIGHT JOIN: inverse of a LEFT JOIN.
- FULL OUTER JOIN: returns everything if there's a match on either side.



Diagrams from w3schools.com

Re-create *orig_data*

- Using many JOINs and *HomeValueIndex* try to get back all of the data that is stored in orig_data…

Views

- Views are stored SELECT statements that act like tables.
- Queries are performed on the underlying tables so views update as the underlying data is updated.

```
CREATE [TEMPORARY] VIEW view_name (col1,col2,...) AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- Let's put together a New England view (*ne*_view) that mimics the *ne_data* table.

BOSTON UNIVERSITY

The optional TEMPORARY keyword makes the view last for the duration of the query.

Indexing

- *HomeValueIndex* has 6,133,491 rows. Queries take a few seconds to execute.
  - A basic query will likely involve using the views that perform many joins, so a lot of data matching is going on.
- A SQL **index** is a high-speed lookup that's added to a column or columns of a table to improve query speeds.

```
CREATE INDEX index_name ON table_name
( col1, col2, …) ;
```

- Indexes are automatically updated when data is added, changed, or deleted from columns involved in the index.

BOSTON UNIVERSITY

Speed Test

- Run the SELECT, then add an index, then re-run.

```
SELECT HomeValueIndex.State as StateID, States.State
FROM HomeValueIndex
JOIN States ON HomeValueIndex.State=States.rowid  ;
```

```
CREATE INDEX hvi_state_index ON
HomeValueIndex ( State ) ;
```

BOSTON UNIVERSITY

Add a multi-column index to *HomeValueIndex*

- First do a SELECT COUNT(*) from the *ne_view* view and check the time for it to execute.
- To build the index let's use the DB Browser GUI.
- Repeat the query on *ne_view*. Any difference in query time?

BOSTON UNIVERSITY

Indexing downsides

- Indexes take up storage space.
- They need to be updated when data is insert, updated, or deleted
  - This can **significantly** slow down these operations.
- They should be used with some care.
- Some DBMS software (like SQL Server's GUI system) can suggest query optimizations which may involve adding indexes to tables.

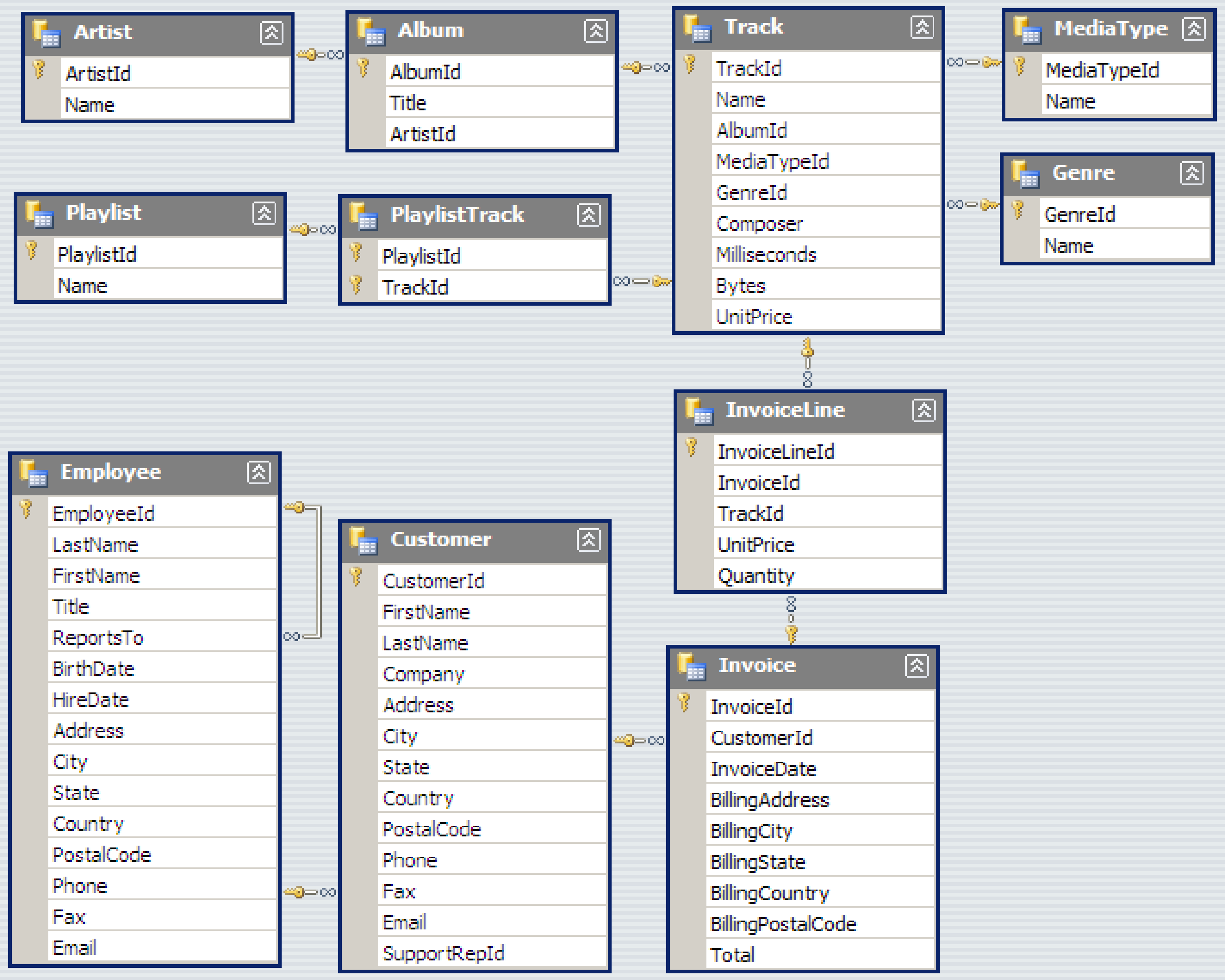
Homework

- Let’s take a look at the Chinook database (chinook.db)
- This is a simulated online music store (like iTunes).
- Here’s the layout of the database…
- Your homework will involve writing several SQL queries to extract data from the database.

BOSTON UNIVERSITY

Chinook table diagram

symbol indicates a *foreign key*



BOSTON UNIVERSITY