Spring 2024 CSE 380

1-23-2024

Quiz Info

- We had our first quiz today
- Quiz 2 is moved to Feb 6th, the syllabus is updated
- I won't be returning quizzes, you can view a quiz in my office

Review

WHERE Clauses

- SELECT ... FROM ... WHERE ...;
- The where clause stipulates a test (predicate) that each row must pass to be returned in the select statement.

SELECT title, genre FROM Movies WHERE length < 120;

WHERE Clauses

SELECT title, genre FROM Movies WHERE length < 120;

```
sqlite> SELECT title, genre FROM Movies WHERE length < 120;
title          genre
-------------  ----------
Wayne's World  comedy
Back to the F  adventure
```

Between

SELECT title, genre FROM Movies WHERE length BETWEEN 90 AND 130;

note: between is inclusive of the bounds

```
sqlite> SELECT title, genre FROM Movies WHERE length BETWEEN 90 AND 130;
title       genre
----------  ----------
Star Wars   sciFi
Wayne's Wo  comedy
Back to th  adventure
```

IS and IS NOT

SELECT title, genre FROM Movies WHERE length IS NULL;

'IS' operator is only to be used to test for NULL values

```
sqlite> SELECT title, genre FROM Movies WHERE length is NULL;
title              genre
-----------------  ----------
Good Will Hunting  drama
```

IN

SELECT title, genre FROM Movies WHERE length IN(90,130);

- IN tests for inclusion in a static, parenthesized list

```
sqlite> SELECT title, genre FROM Movies WHERE length IN (90, 130);
sqlite>
```

Comparing NULLS to Values

- Comparing any value (including NULL itself) with NULL yields UNKNOWN.
- A tuple is in a query answer if and only if the WHERE clause is TRUE (not FALSE or UNKNOWN).

SQL Column Constraints

- Unique:
  - Used to specify a column whose values will always be unique for every row.
  - CREATE TABLE students (msu_id TEXT UNIQUE,
    pid INTEGER UNIQUE,
    first_name TEXT);

  The DBMS will raise an error if the constraint is violated.

NOT NULL

- Used to designate columns that may not be NULL

  CREATE TABLE students (msu_id TEXT UNIQUE NOT NULL, section INTEGER NOT NULL);

- Raises error if a NULL value is inserted.

Primary key

- Each table can have at most one PRIMARY KEY.
  - It is implicitly NOT NULL and UNIQUE
  - It is used to specify a specific column in the table

```
CREATE TABLE students (msu_id TEXT PRIMARY KEY,
                       grade REAL);
```

Joins

- Often, queries need to combine (join) rows from multiple tables.
- Joins specify which rows get merged with rows from a second table.
- As there are many possible methods to match rows together, there are many types of joins.
- You will need to know (memorize) the different joins.
- Annoying, but very useful knowledge.

Joins

professors

id	name
esfahanian	Abdol
mariani	James
dyksen	Wayne
NULL	demo-prof

```
CREATE TABLE professors (id TEXT, name TEXT);
INSERT INTO professors VALUES('esfahanian','Abdol');
INSERT INTO professors VALUES('mariani','James');
INSERT INTO professors VALUES('dyksen','Wayne');
INSERT INTO professors VALUES(NULL, 'demo-student');
```

```
CREATE TABLE associates (responder_id TEXT, associate_id TEXT);
INSERT INTO associates VALUES('mariani','dyksen');
INSERT INTO associates VALUES('mariani','esfahanian');
INSERT INTO associates VALUES('dyksen','mariani');
INSERT INTO associates VALUES('esfahanian','dyksen');
```

associates

responder_id	associate_id
mariani	dyksen
mariani	esfahanian
dyksen	mariani
esfahanian	dyksen

Inner Join

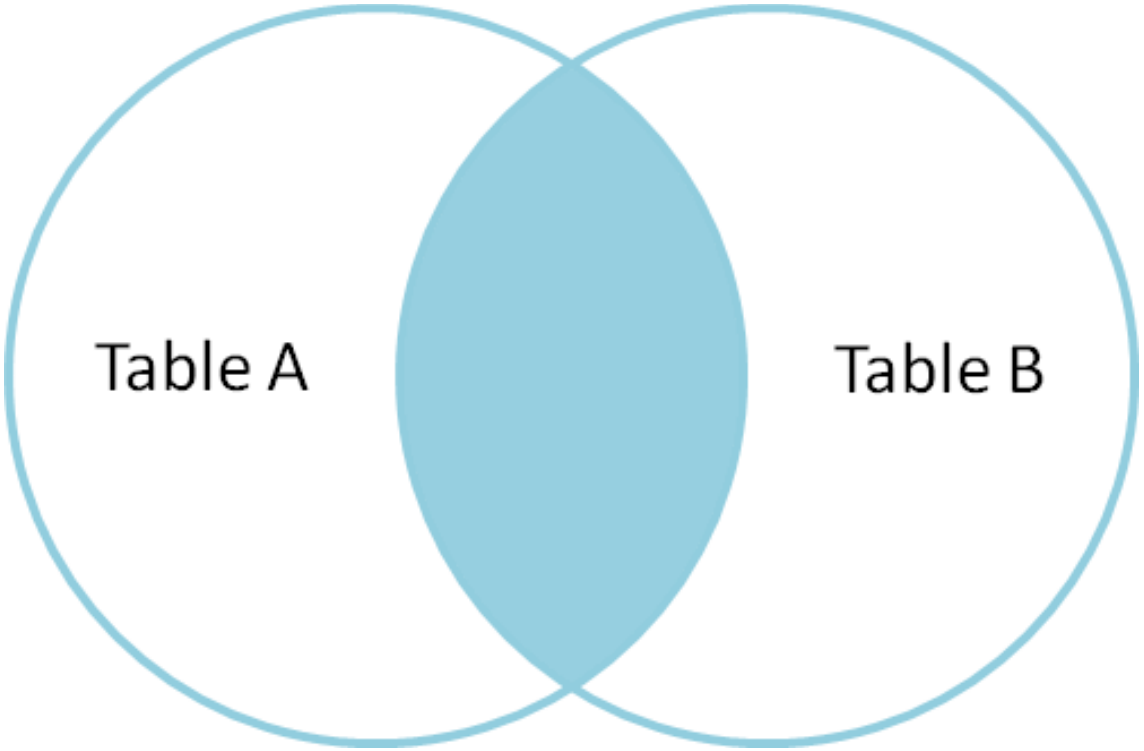
- INNER JOIN is like the CROSS JOIN, except only rows that match a predicate are allowed. Frequently that predicate includes primary keys to match associated data

```
SELECT * FROM professors INNER JOIN
associates
    ON id = responder_id;
```

- You can equivalently do an implicit inner join and use WHERE to filter rows:

```
SELECT * FROM students, associates
    WHERE id = responder_id;
```

Inner Join



Retuned rows are those that match both tables (shaded).

Inner Join

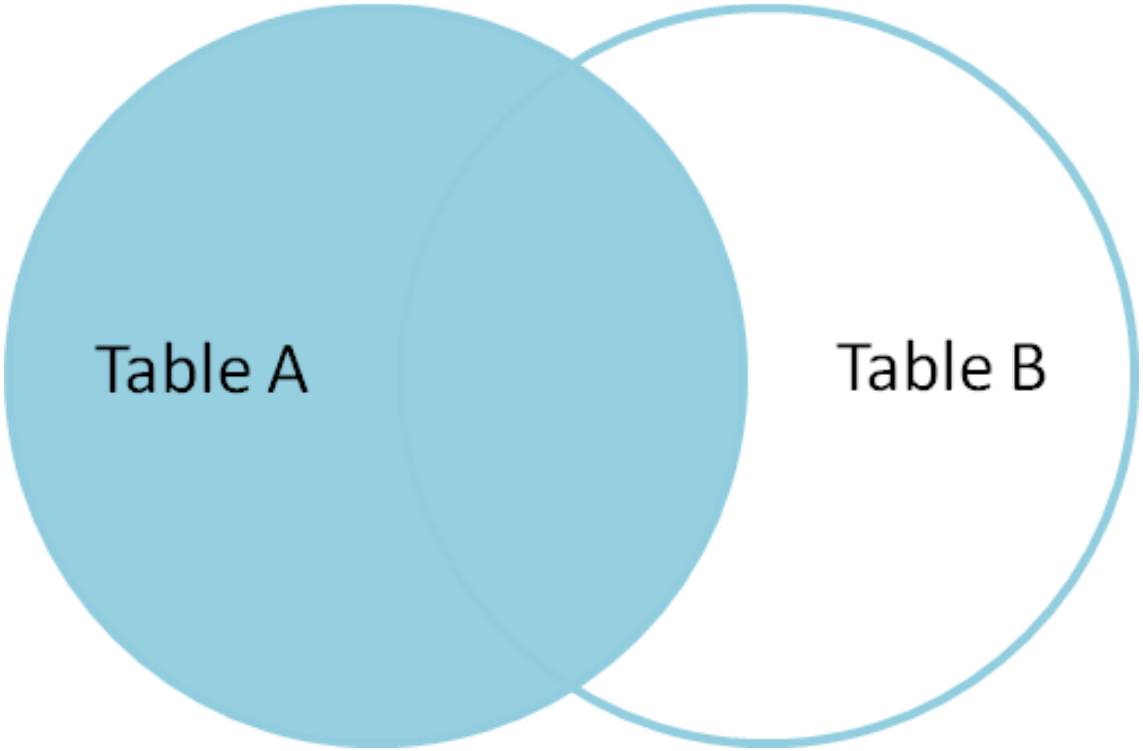
- SELECT * FROM professors INNER JOIN associates ON id = responder_id;

```
sqlite> SELECT * FROM professors INNER JOIN associates ON id = responder_id;
id          name        responder_id  associate_id
----------  ----------  ------------  ------------
esfahanian  Abdol       esfahanian    dyksen
mariani     James       mariani       dyksen
mariani     James       mariani       esfahanian
dyksen      Wayne       dyksen        mariani
```

Left Outer Join

- LEFT OUTER JOIN includes all the rows to the left of the JOIN keyword, where there is a match, it add the matching column data to rows. If there isn't a match NULL values are used instead.
- SELECT * FROM professors LEFT OUTER JOIN associates
  ON id = responder_id;
- Note: order matters!!!
- SELECT * FROM associates LEFT OUTER JOIN professors
  ON id = responder_id;

Left Outer Join



Left Outer Join

SELECT * FROM professors LEFT OUTER JOIN associates ON id = responder_id;

```
[sqlite> SELECT * FROM professors LEFT OUTER JOIN associates ON id=responder_id;
id          name        responder_id  associate_id
----------  ----------  ------------  ------------
esfahanian  Abdol       esfahanian    dyksen
mariani     James       mariani       dyksen
mariani     James       mariani       esfahanian
dyksen      Wayne       dyksen        mariani
            demo-stude
```

New Material

SQL Union

- UNION combines the result sets of two queries.
- Column data types in the two queries must match.
- UNION combines by column position rather than column name.
- If we wanted all the names of Artists and Albums:

```
SELECT Artist.Name
  FROM Artist
UNION
SELECT Album.Title
  FROM Album;
```

SQL Union

```
sqlite> SELECT Artist.Name FROM Artist WHERE Artist.Name IN (SELECT Album.Title FROM Album) ORDER BY Artist.Name DESC LIMIT 10;
Name
----------
Van Halen
The Doors
Temple of
Raul Seixa
Pearl Jam
Olodum
Iron Maide
Body Count
Black Sabb
Audioslave
```

```
SELECT Artist.Name
  FROM Artist
UNION
SELECT Album.Title
  FROM Album;
```

SQL Union All

- UNION performs a DISTINCT on the result set, eliminating any duplicate rows.
- UNION ALL does not remove duplicates, and it therefore faster to perform than UNION.
- If I wanted the Artists and Album names, but duplicates were okay:

```
SELECT Artist.Name
  FROM Artist
UNION ALL
SELECT Album.Title
  FROM Album;
```

When is a Left Outer Join the Same as an Inner Join?

- Never
- When the left rows always have a match
- When the left rows don't have NULLs
- Always

SQL Subqueries

- A subquery is a SQL query within a query.
- Subqueries are nested queries that provide data to the enclosing query.
- Subqueries can return individual values or a list of records
- Subqueries must be enclosed with parenthesis
- If I wanted all the Artists who released an Album with just their own name:

```
SELECT Artist.Name
FROM Artist
WHERE Artist.Name IN (SELECT Album.Title FROM
Album);
```

SQL Subqueries

- Subqueries often used to perform tests for set membership, make set comparisons, and determine set cardinality

```
SELECT name FROM courses WHERE
  semester = 'Fall' AND semester = 'Spring';
```

```
SELECT name FROM courses WHERE semester
  = 'Fall' AND name in (SELECT name FROM
  courses WHERE semester = 'Spring');
```

name	semester
CSE 480	Fall
CSE 480	Spring
CSE 498	Fall
CSE 498	Spring
CSE 422	Fall
CSE 476	Fall

```
sqlite> SELECT name FROM courses WHERE semester = 'Fall' AND semester = 'Spring';
sqlite>
```

```
sqlite> SELECT name FROM courses WHERE semester = 'Fall' AND name IN (SELECT name FR
OM courses WHERE semester = 'Spring');
name
----------
CSE 480
CSE 498
sqlite>
```

Rest of SQL SELECT

- MIN MAX
  - Select min return the minimum value for a column
  - Select max return the maximum value for a column

SELECT first_name, MIN(siblings) FROM students;

```
sqlite> SELECT first_name, MIN(siblings) FROM students;
first_name  MIN(siblings)
----------  -------------
Ishita      0
```

SELECT first_name, MAX(siblings) FROM students;

```
sqlite> SELECT first_name, MAX(siblings) FROM students;
first_name  MAX(siblings)
----------  -------------
Odon        8
```

COUNT, SUM, TOTAL, AVG

- COUNT returns a count of non-null values
- SUM returns the sum of the data (NULL if all rows are NULL)
- TOTAL returns total sum (NULL is zero)
- AVG returns average of data values

```
sqlite> SELECT COUNT(siblings) FROM students;
COUNT(siblings)
---------------
110
```

```
sqlite> SELECT SUM(siblings) FROM students;
SUM(siblings)
-------------
180
```

```
sqlite> SELECT TOTAL(siblings) FROM students;
TOTAL(siblings)
---------------
180.0
```

```
sqlite> SELECT AVG(siblings) FROM students;
AVG(siblings)
----------------
1.63636363636364
```

DISTINCT

- SELECT DISTINCT returns only distinct (different) values.
- SELECT DISTINCT eliminates duplicate records from the results.
- DISTINCT can be used with aggregates: COUNT, AVG, MAX, etc.
- DISTINCT operates on a single column. DISTINCT for multiple columns is not supported.

SELECT DISTINCT siblings FROM students;

```
sqlite> SELECT DISTINCT siblings FROM students;
siblings
---------
1
0
2
3
5
8
4
6
```

SELECT count(DISTINCT siblings) FROM students;

```
sqlite> SELECT COUNT(DISTINCT siblings) FROM students;
COUNT(DISTINCT siblings)
------------------------
8
```

GROUP BY

- The GROUP BY clause groups records into summary rows.
- GROUP BY returns one records for each group.
- GROUP BY typically also involves aggregates: COUNT, MAX, SUM, AVG, etc.
- GROUP BY can group by one or more columns.

SELECT count(), siblings FROM students GROUP BY siblings;

```
sqlite> SELECT count(), siblings FROM students GROUP BY siblings;
count()     siblings
----------  ----------
1
13          0
43          1
38          2
11          3
1           4
2           5
1           6
1           8
```

LIMIT

- Last clause of SELECT statement.
- Limits the number of rows to the value (or less).
- Often useful with ORDER BY to get extremes.

```
SELECT first_name, siblings FROM students
  ORDER BY siblings DESC LIMIT 10;
```

```
sqlite> SELECT first_name, siblings FROM students ORDER BY siblings DESC LIMIT 10;
first_name  siblings
----------  ----------
Odon        8
Jimmy       6
Yunpeng     5
Frederick   5
Andrew      4
Emily       3
Ajuisiwon   3
Finn        3
Brandon     3
Gregory     3
```

All the parts of a SELECT query

- SELECT column-names
- FROM table-name
- WHERE condition
- GROUP BY column-names
- HAVING condition
- ORDER BY column-names
- LIMIT max-rows
- ;

CRUD

- Four fundamental operations that apply to any database are:
  - Read the data -- *SELECT*
  - Insert new data -- *INSERT*
  - Update existing data -- *UPDATE*
  - Remove data -- *DELETE*
- Collectively these are referred to as **CRUD** (Create, Read, Update, Delete).

INSERT INTO

- The INSERT INTO statement is used to add new data to a database.
- The INSERT INTO statement adds a new record to a table.
- INSERT INTO can contain values for some or all of its columns.
- INSERT INTO can be combined with a SELECT to insert records.

```
INSERT INTO spelling_team, (first_name, spelling)
  VALUES ('James', 10);
```

INSERT Multiple Values

- You can insert multiple rows with a single INSERT statement:

```
INSERT INTO spelling_team (first_name, spellings)
VALUES ('James', 10),
       (Abigail', 9);
```

INSERT INTO SELECT

- You can also use a SELECT clause to generate the needed rows, but you need to return the correct column types and order.

```
INSERT INTO spelling_team (first_name, spelling)
SELECT students.first_name, students.spelling
FROM students ORDER BY spelling DESC LIMIT 10;
```

```
sqlite> SELECT * FROM spelling_team;
first_name  spelling
----------  ----------
Oscar       12
Yujin       10
Shafkat     10
Rohit       10
Matthew     10
Marla       10
Ishita      10
Elio        10
David       10
Brandon     10
```

UPDATE

- The UPDATE statement updates data values in a database.
- UPDATE can update one or more records in a table.
- Use the WHERE clause to UPDATE only specific records.

```
UPDATE students SET spelling = 0 WHERE spelling >10;
```

```
sqlite> SELECT * FROM spelling_team;
first_name  spelling
----------  ----------
Yujin       10
Shafkat     10
Rohit       10
Matthew     10
Marla       10
Ishita      10
Elio        10
David       10
Brandon     10
Antonio     10
```

UPDATE

- The UPDATE statement updates data values in a database.
- UPDATE can update one or more records in a table.
- Use the WHERE clause to UPDATE only specific records.

```
UPDATE students SET spelling = 0 WHERE spelling >10;
UPDATE students SET fav_word = fav_word || '!';

|| is concatenate in SQL
```

DELETE

- DELETE permanently removes records from a table.
- DELETE can delete one or more records in a table.
- Use the WHERE clause to DELETE only specific records.

```
DELETE FROM spelling_team; -- removes all rows

DELETE FROM spelling_team WHERE got_answer_right = 'False';
--removes some rows
```

AUTOINCREMENT

- SQLite AUTOINCREMENT is a keyword used for auto incrementing a value of a field in the table. We can auto increment a field value by using AUTOINCREMENT keyword when creating a table with specific column name to auto incrementing it.
- The keyword AUTOINCREMENT can be used with a INTEGER PRIMARY KEY field only.
- CREATE TABLE table_name (column1 INTEGER PRIMARY KEY AUTOINCREMENT, column2 datatype, column3 datatype, ... columnN datatype);

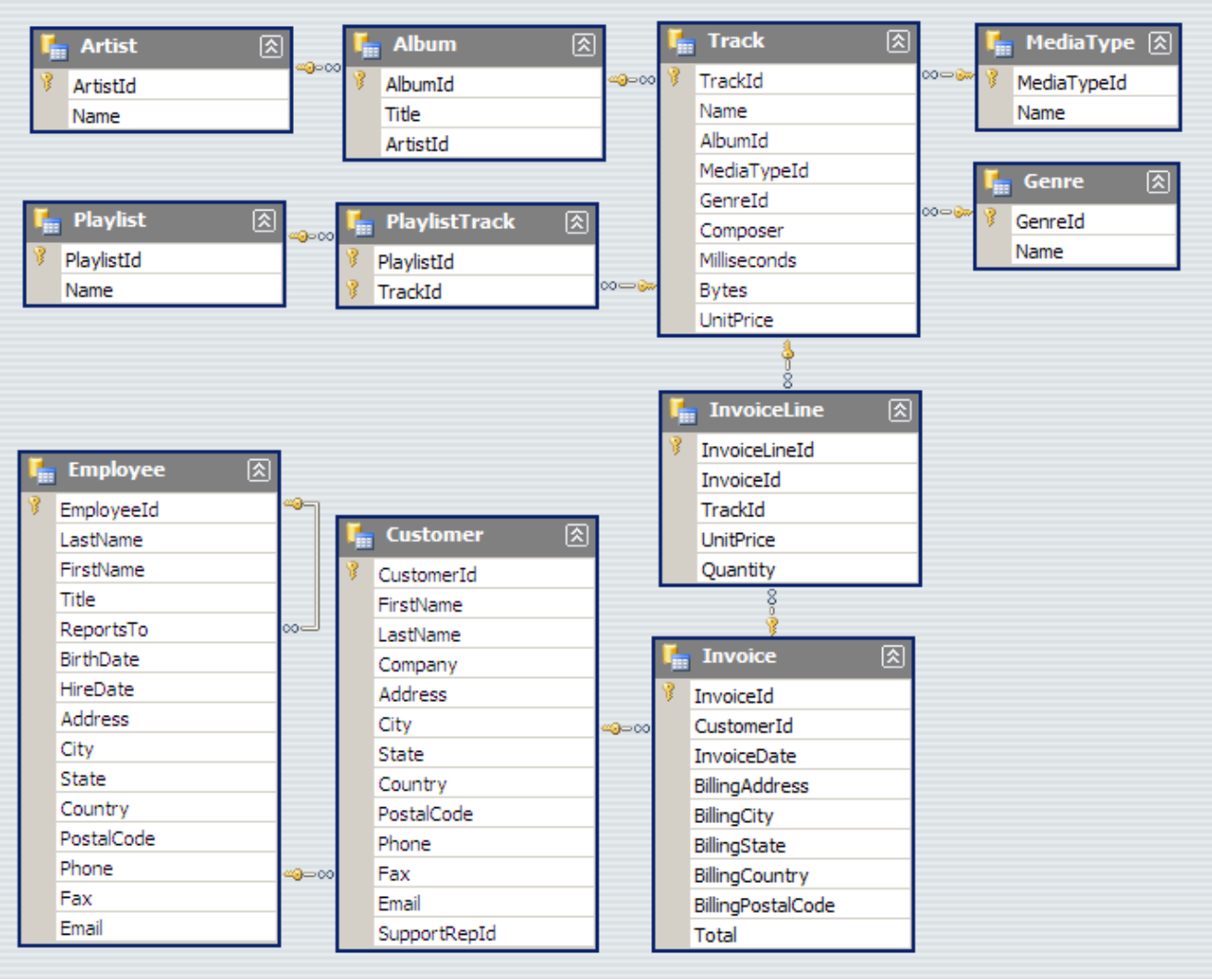
Should you use it?

- From https://www.sqlite.org/autoinc.html:

  "The AUTOINCREMENT keyword imposes extra CPU, memory, disk space, and disk I/O overhead and should be avoided if not strictly needed. It is usually not needed."
- If a column is INTEGER PRIMARY KEY, it already will autofill a unique value if a value isn't provided.
  - The only difference is AUTOINCREMENT guarantees monotonically increasing values, instead of just unique.

Practice

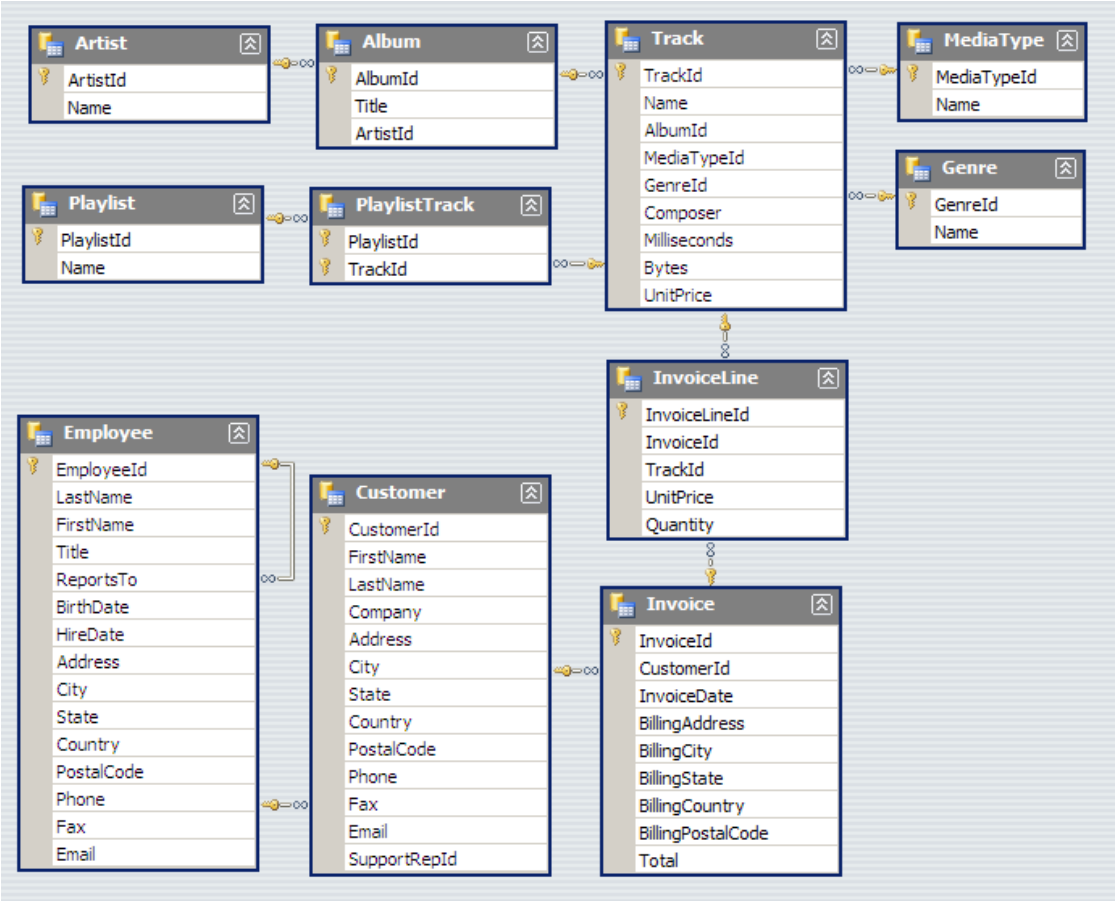
Chinook Dataset



Test Databases

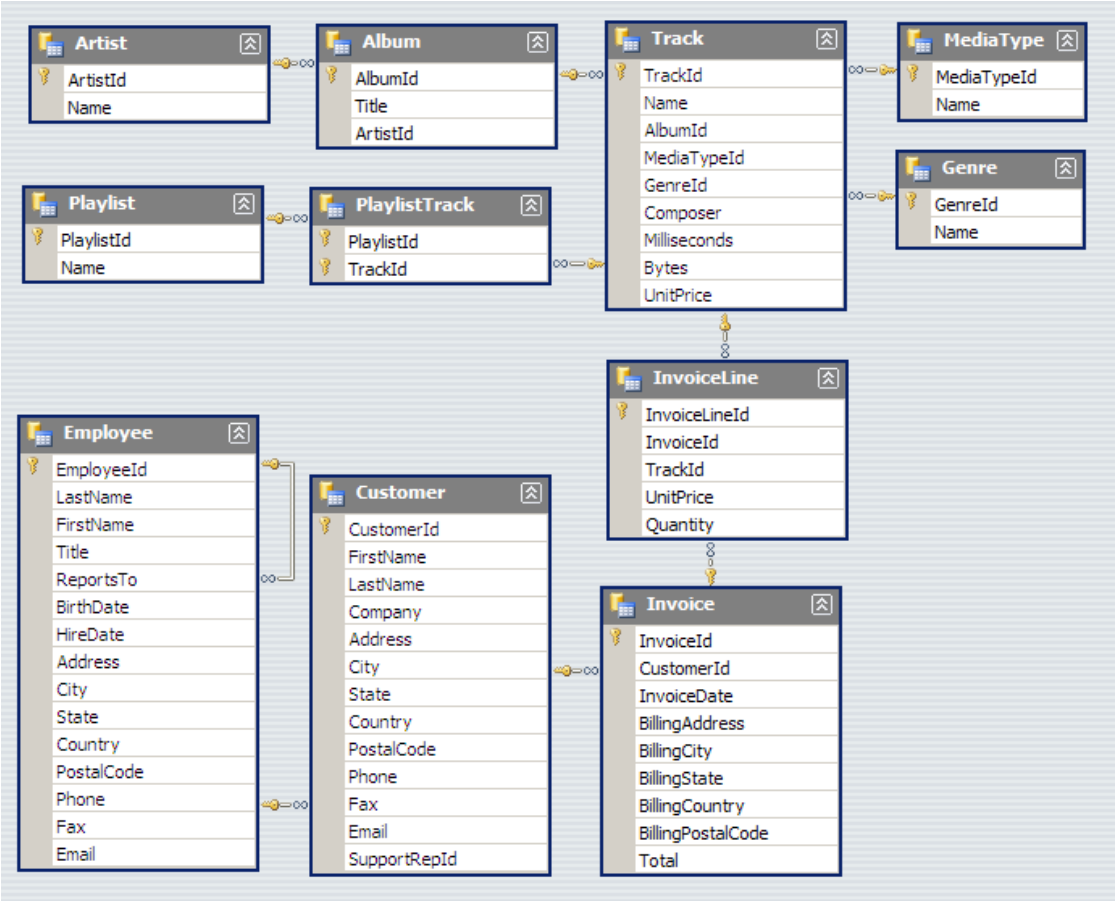
- Chinook Music Database
  - Used for examples
- Create chinook DB instance - .sql file on D2L

sqlite3 chinook.db < Chinook_Sqlite.sql



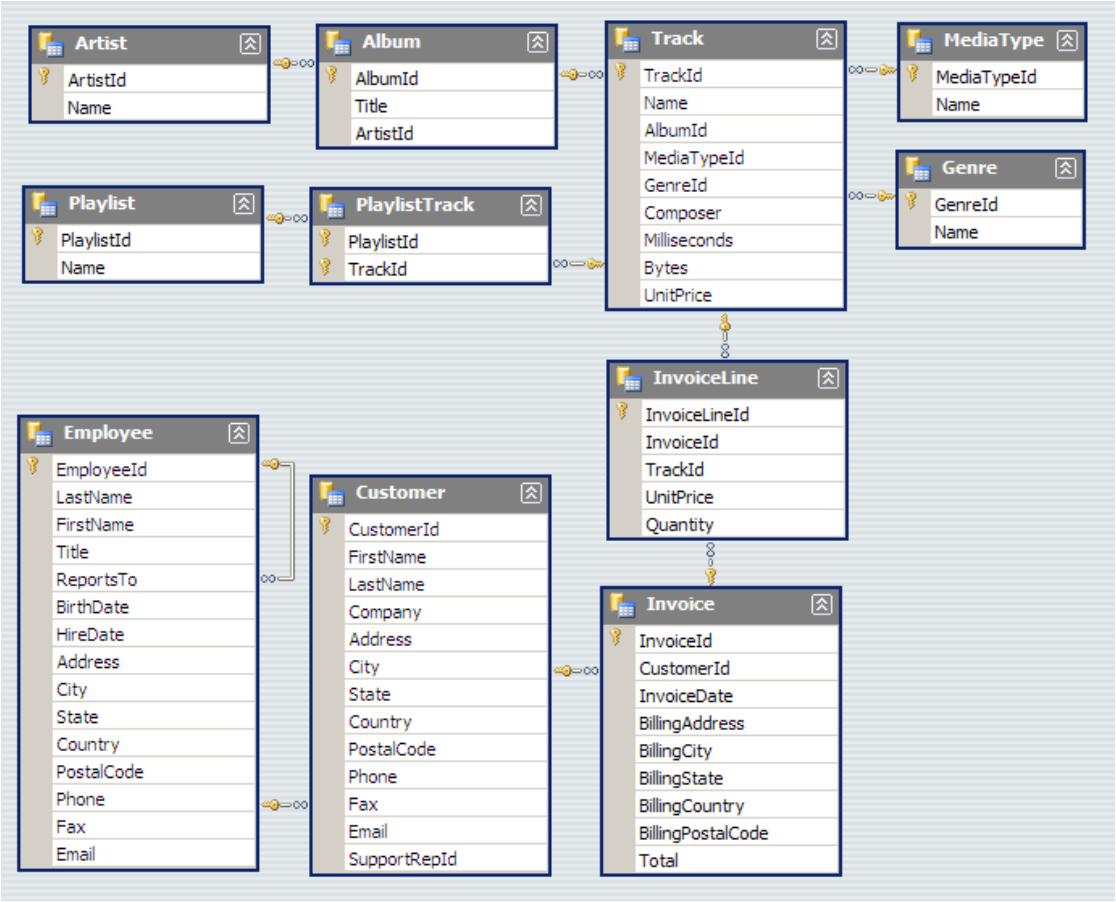
Practice

Provide a query that shows the total sales per country in order from most to least sales



Practice

Provide a query that shows the number of customers assigned to each sales agent



Practice

Provide a query that shows the total number of tracks in each playlist, and includes the playlist name in the result

That's it for today

- Questions?