Data Engineering in the Cloud

Basic SQL Queries - Part II

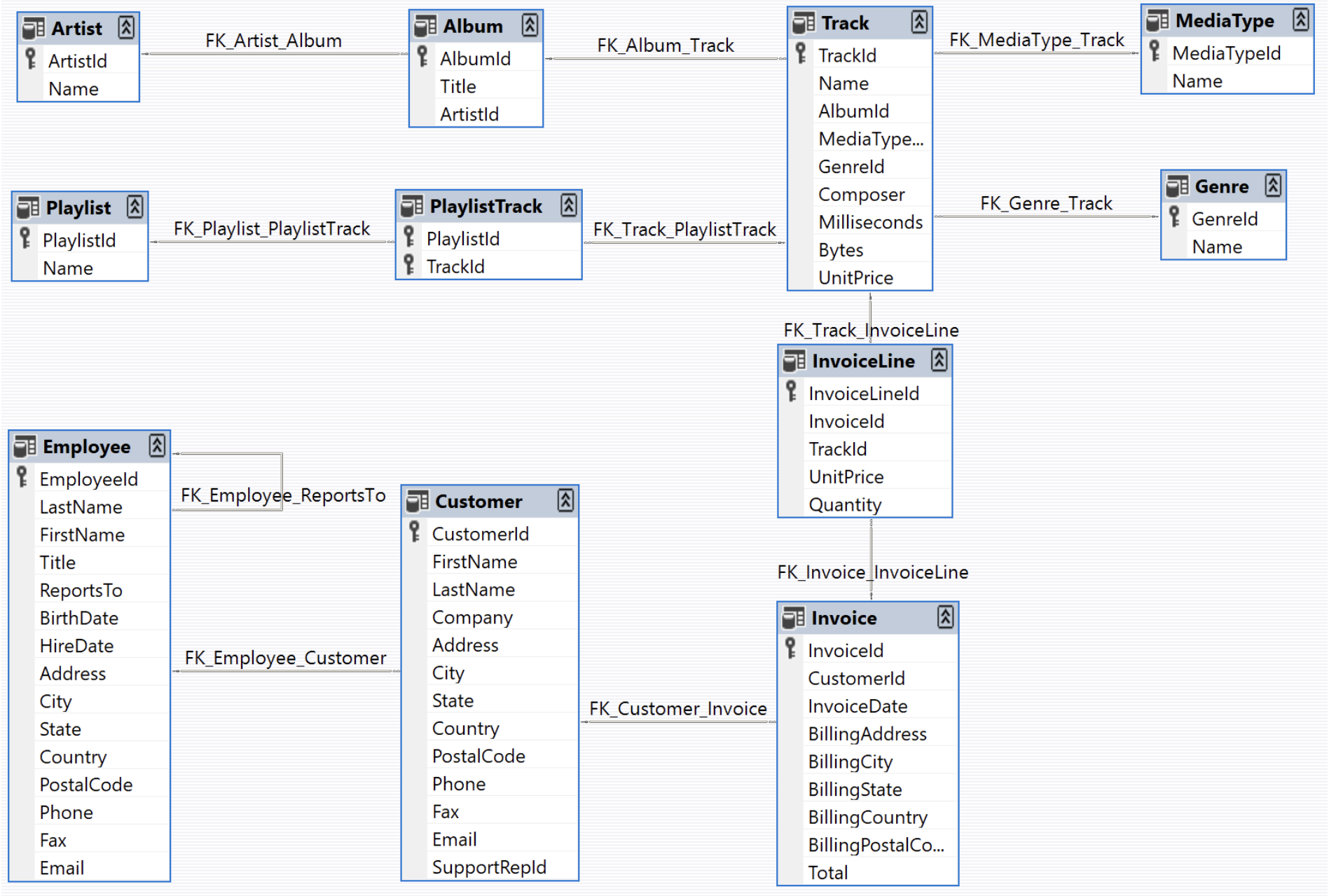
Xuemao Zhang
East Stroudsburg University

January 18, 2025

Outline

- chinook-database
- Retrieving data from multiple tables
- Joining tables
- SQL Join Expressions
  - INNER JOIN
  - LEFT JOIN
  - RIGHT JOIN
  - FULL JOIN
  - CROSS JOIN
- Creating a view

chinook database



Retrieving data from multiple tables

- List each required table in the **FROM** clause
- **SELECT** and **WHERE** clauses can refer to attributes of any of these tables
- In general, `conditions` (such as two attribute values being equal) need to 'connect' the tables

Joining tables

- Consider the `Album` and `Track` tables

```
SELECT DISTINCT AlbumId FROM Album a ;
```

- Combine the two tables:

```
SELECT *
from Album a, Track t
where a.AlbumId = t.AlbumId;
```

```
SELECT *
from Album a, Track t
where a.AlbumId = t.AlbumId
and t.AlbumId = 34;
```

- We can join more than two tables in this way

SQL Join Expressions

- Instead of 'connecting' the tables in the **WHERE** clause, we can use an explicit **JOIN** in the **FROM** clause.
- Types of joins
  - INNER JOIN
  - LEFT JOIN
  - RIGHT JOIN
  - FULL OUTER JOIN
  - CROSS JOIN
- All joins in the last slide are essentially `INNER JOIN`s

SQL Join Expressions - INNER JOIN

- `INNER JOIN`: The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in both tables.
  - INNER JOIN animation
- SQL INNER JOIN Syntax

```
SELECT      column-name(s)
FROM        table1
INNER JOIN  table2
ON          table1.column-name=table2.column-name;
```

- **Note**: `INNER JOIN` can be replaced with `JOIN`

SQL Join Expressions - INNER JOIN

```
SELECT *
FROM Album a
JOIN Track t
ON a.AlbumId = t.AlbumId
where t.AlbumId = 34;
```

SQL Join Expressions - LEFT JOIN

- `LEFT JOIN`: The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.
  - LEFT JOIN animation
- SQL LEFT JOIN Syntax

```
SELECT      column-name(s)
FROM        table1
LEFT JOIN   table2
ON          table1.column-name=table2.column-name;
```

- **Note**: In some databases `LEFT JOIN` is called `LEFT OUTER JOIN`.

SQL Join Expressions - LEFT JOIN

```
SELECT *
FROM Album a
LEFT JOIN Track t
ON a.AlbumId = t.AlbumId
where t.AlbumId = 34;
```

- If we interchange the order of `Track` and `Album`,

```
SELECT *
FROM Track t
LEFT JOIN Album a
ON a.AlbumId = t.AlbumId
where t.AlbumId = 34;
```

SQL Join Expressions - RIGHT JOIN

- `RIGHT JOIN`: The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.
  - RIGHT JOIN animation
- SQL RIGHT JOIN Syntax

```
SELECT      column-name(s)
FROM        table1
RIGHT JOIN  table2
ON          table1.column-name=table2.column-name;
```

- **Note**: In some databases `RIGHT JOIN` is called `RIGHT OUTER JOIN`.

SQL Join Expressions - RIGHT JOIN

```
SELECT *
FROM Album a
RIGHT JOIN Track t
ON a.AlbumId = t.AlbumId
where t.AlbumId = 34;
```

- It is equivalent to

```
SELECT *
FROM Track t
LEFT JOIN Album a
ON a.AlbumId = t.AlbumId
where t.AlbumId = 34;
```

SQL Join Expressions - FULL JOIN

- `FULL JOIN`: The FULL JOIN keyword returns all rows from the left table (table1) and from the right table (table2). The FULL JOIN keyword combines the result of both LEFT and RIGHT joins.
  - FULL JOIN animation
- SQL FULL JOIN Syntax

```
SELECT      column-name(s)
FROM        table1
FULL JOIN   table2
ON          table1.column-name=table2.column-name;
```

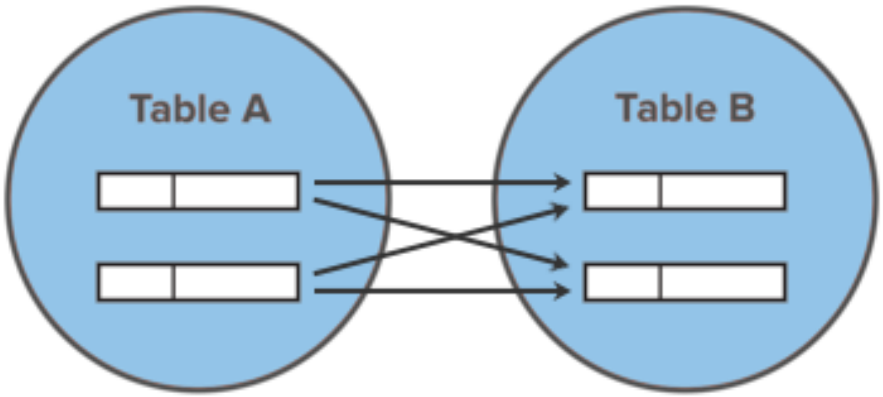
- **Note**: The `FULL JOIN` generally is written as `FULL OUTER JOIN`.

SQL Join Expressions - FULL JOIN

```
SELECT *
FROM Album a
FULL JOIN Track t
ON a.AlbumId = t.AlbumId
where t.AlbumId = 34;
```

SQL Join Expressions - CROSS JOIN

- `CROSS JOIN`: The CROSS JOIN keyword produces a Cartesian product of rows in the two tables.



SQL CROSS JOIN

- SQL CROSS JOIN Syntax

```
SELECT      column-name(s)
FROM        table1
CROSS JOIN  table2;
```

SQL Join Expressions - CROSS JOIN

```
SELECT *
FROM Album a
CROSS JOIN Track t
where  a.AlbumId = 34;
```

Creating a view

- A view in a database is a virtual table that is based on the result set of a SELECT query.
  - It derives its data from one or more underlying tables through a SELECT query.
  - It does not store data physically
- They can present only the necessary columns or rows to the user, hiding the complexity of the underlying data structure.
- Users can be granted access to the view without giving them direct access to the underlying tables.
- Views can perform computations, aggregate data, and provide summarized results.

Creating a view

- Create a view to list albums with their artist names

```
CREATE VIEW AlbumArtistView AS
SELECT
    a.AlbumId,
    a.Title AS AlbumTitle,
    ar.Name AS ArtistName
FROM
    Album a
JOIN
    Artist ar ON a.ArtistId = ar.ArtistId;

SELECT * from AlbumArtistView;
```

License