HBSA – DSSB SQL Workshop Fall 16

Lecturers: Jerry Chen and Dhruv Relwani

TA's: Ricky Pan and Cassie Zhang

Workshop Structure

- We will be going back in forth between slides to demonstrate concepts in Sqlite Studio.
- You are encouraged to follow along on your own computers.

- Topics Covered:

Tables	Basic Syntax	Aggregate Functions
Other Useful Statements	Relational Databases	Joins

Goals and Objectives

1. Give you hands-on experience with the SQL language
2. Understand practical uses of SQL
3. Enough knowledge to practice on your own

What is SQL

- A popular querying language to manipulate data in databases
- “Structured Query Language”

What is SQL

- A popular querying language to manipulate data in databases
- “Structured Query Language”

What is SQL

- A popular querying language to manipulate data in databases
- “Structured Query Language”

What is SQL

- A popular querying language to manipulate data in databases
- “Structured Query Language”

What is SQL

- A popular querying language to manipulate data in databases
- “Structured Query Language”

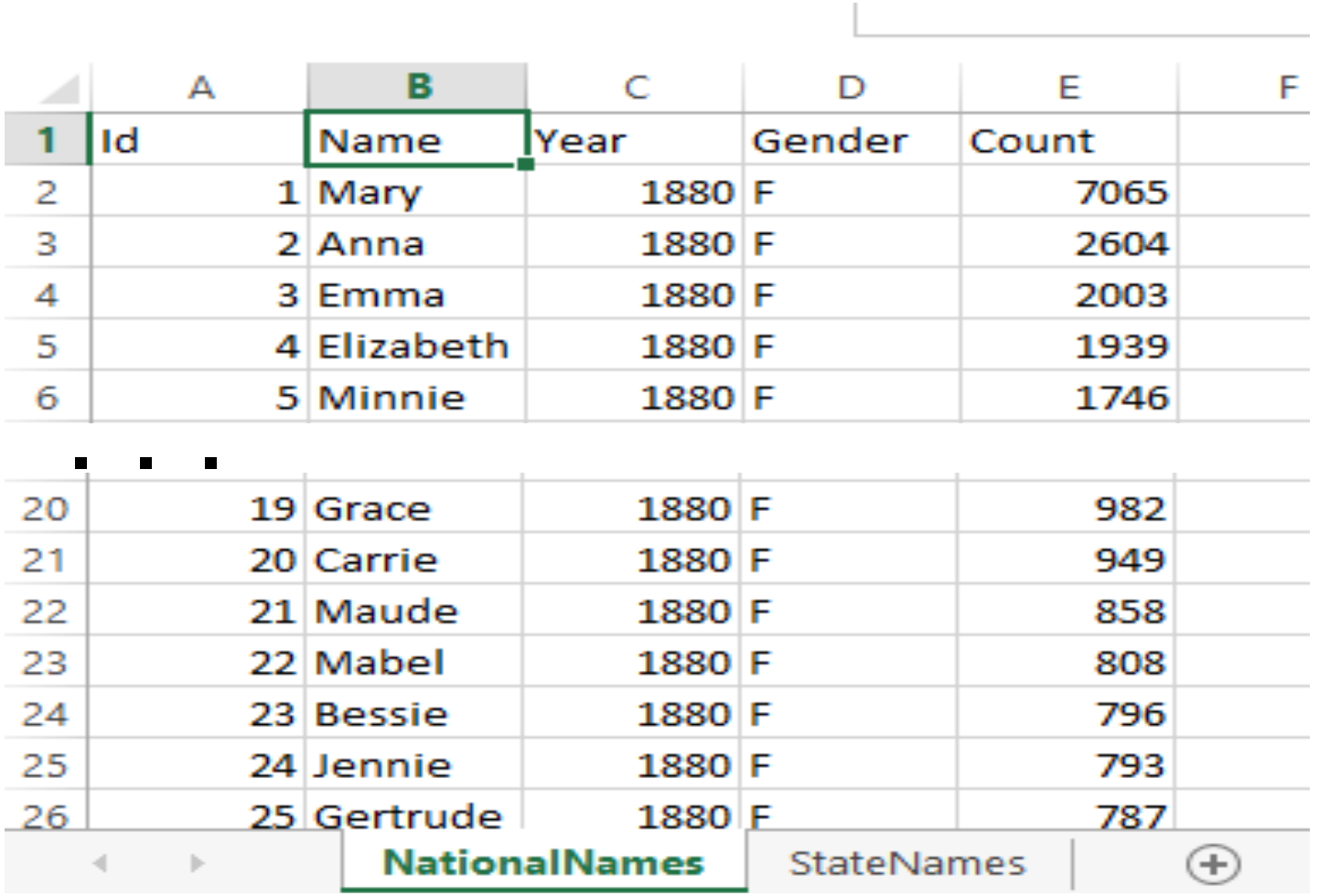
Tables

- These are an “array” of data
- Think Excel spreadsheets

	A	B	C	D	E	F
1	Id	Name	Year	Gender	Count	
2	1	Mary	1880	F	7065	
3	2	Anna	1880	F	2604	
4	3	Emma	1880	F	2003	
5	4	Elizabeth	1880	F	1939	
6	5	Minnie	1880	F	1746	

Tables

- These are an “array” of data
- Think Excel spreadsheets



	A	B	C	D	E	F
1	Id	Name	Year	Gender	Count	
2	1	Mary	1880	F	7065	
3	2	Anna	1880	F	2604	
4	3	Emma	1880	F	2003	
5	4	Elizabeth	1880	F	1939	
6	5	Minnie	1880	F	1746	
...						
20	19	Grace	1880	F	982	
21	20	Carrie	1880	F	949	
22	21	Maude	1880	F	858	
23	22	Mabel	1880	F	808	
24	23	Bessie	1880	F	796	
25	24	Jennie	1880	F	793	
26	25	Gertrude	1880	F	787	

NationalNames StateNames

Basic Syntax

- SELECT
- FROM
- WHERE
- LIMIT

- SELECT DISTINCT
- AS (for shorthand naming purposes)
- SELECT COUNT(*) (to see how many records a table has)

Aggregate Functions

- COUNT()
- SUM()
- AVG()
- What is the average track length? →

```
SELECT AVG(Milliseconds)
FROM Track;
```

Aggregate Functions and the Group By

- COUNT()
- SUM()
- AVG()
- What is the average track length?
- Say you want the number of tracks in each album →

```
SELECT Count(AlbumId)
FROM Track
GROUP BY AlbumId;
```

Group by (Visual)

```
SELECT Count(AlbumId)
FROM Track
GROUP BY AlbumId;
```

Group by (Visual)

```
SELECT Count(AlbumId)
FROM Track
GROUP BY AlbumId;
```

TrackId	Name	AlbumId
1	abc	1
2	def	2
3	ghi	1
4	jkl	3
5	mno	1
6	pqr	2
7	stu	3

Group by (Visual)

```
SELECT Count(AlbumId)
FROM Track
GROUP BY AlbumId;
```

TrackId	Name	AlbumId
1	abc	1
2	def	2
3	ghi	1
4	jkl	3
5	mno	1
6	pqr	2
7	stu	3

TrackId	Name	AlbumId
1	abc	1
3	ghi	1
5	mno	1

TrackId	Name	AlbumId
2	def	2
6	pqr	2

TrackId	Name	AlbumId
4	jkl	3
7	stu	3

Group by (Visual)

```
SELECT Count(AlbumId)
FROM Track
GROUP BY AlbumId;
```

TrackId	Name	AlbumId
1	abc	1
2	def	2
3	ghi	1
4	jkl	3
5	mno	1
6	pqr	2
7	stu	3

TrackId	Name	AlbumId
1	abc	1
3	ghi	1
5	mno	1

3

TrackId	Name	AlbumId
2	def	2
6	pqr	2

2

TrackId	Name	AlbumId
4	jkl	3
7	stu	3

2

Aggregate Functions and the Group By

- COUNT()
- SUM()
- AVG()
- What is the average track length?
- Say you want the number of tracks in each album →

```
SELECT Count(AlbumId)
FROM Track
GROUP BY AlbumId;
```

Aggregate Functions and the Group By

- COUNT()
- SUM()
- AVG()
- What is the average track length?
- Say you want the number of tracks in each album
- Say you want total length of each album →

```
SELECT AlbumId, SUM(Milliseconds)
FROM Track
GROUP BY AlbumId;
```

Other useful Statements

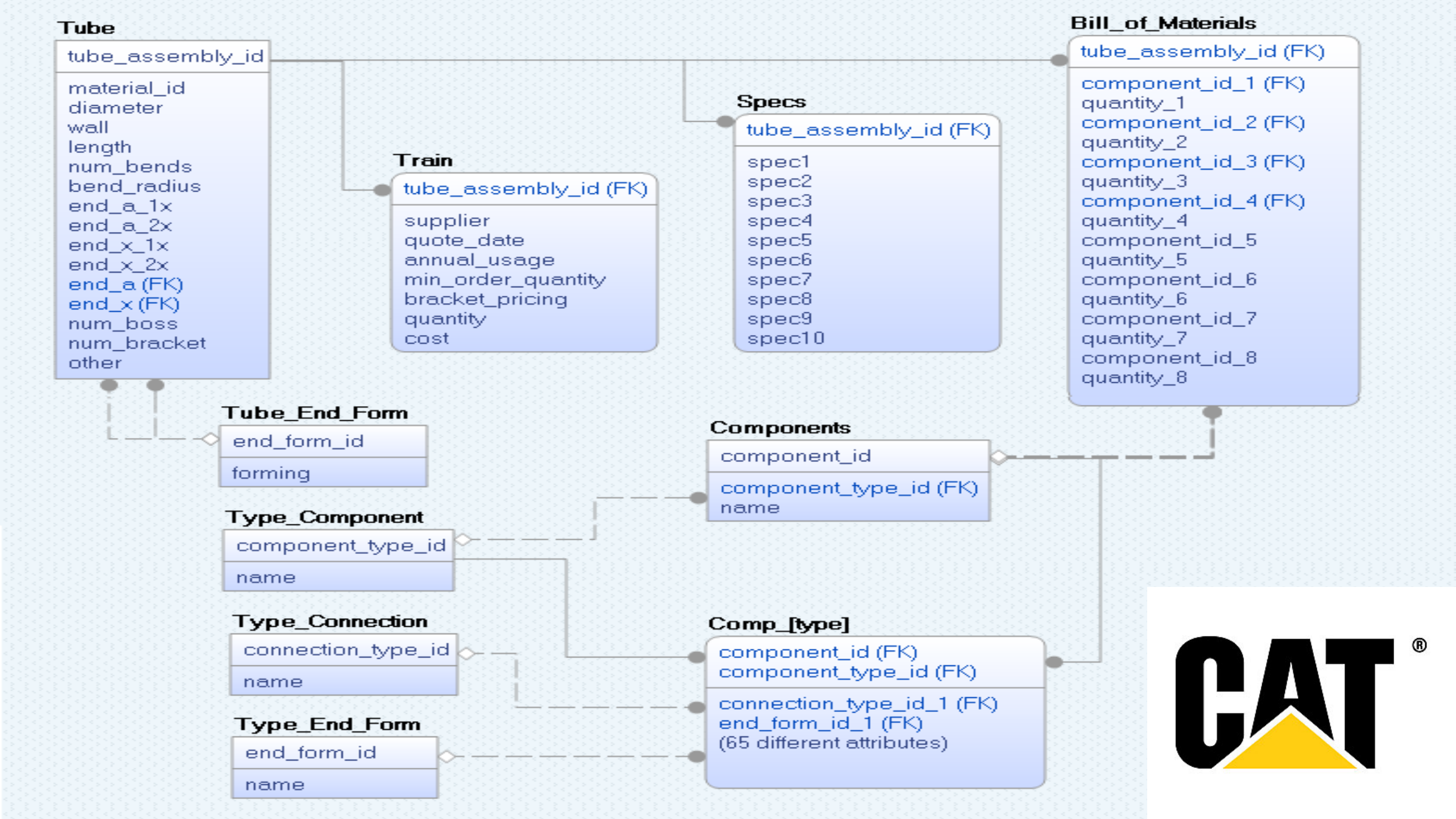
- ORDER BY
  - Default is “ascending”
- Having Statement
  - Very similar to WHERE clause but “Having” is written after (and before ORDER BY)
  - WHERE = Subset data before running the query, no aggregate functions allowed
  - HAVING = Subset data after running query, aggregate functions are allowed

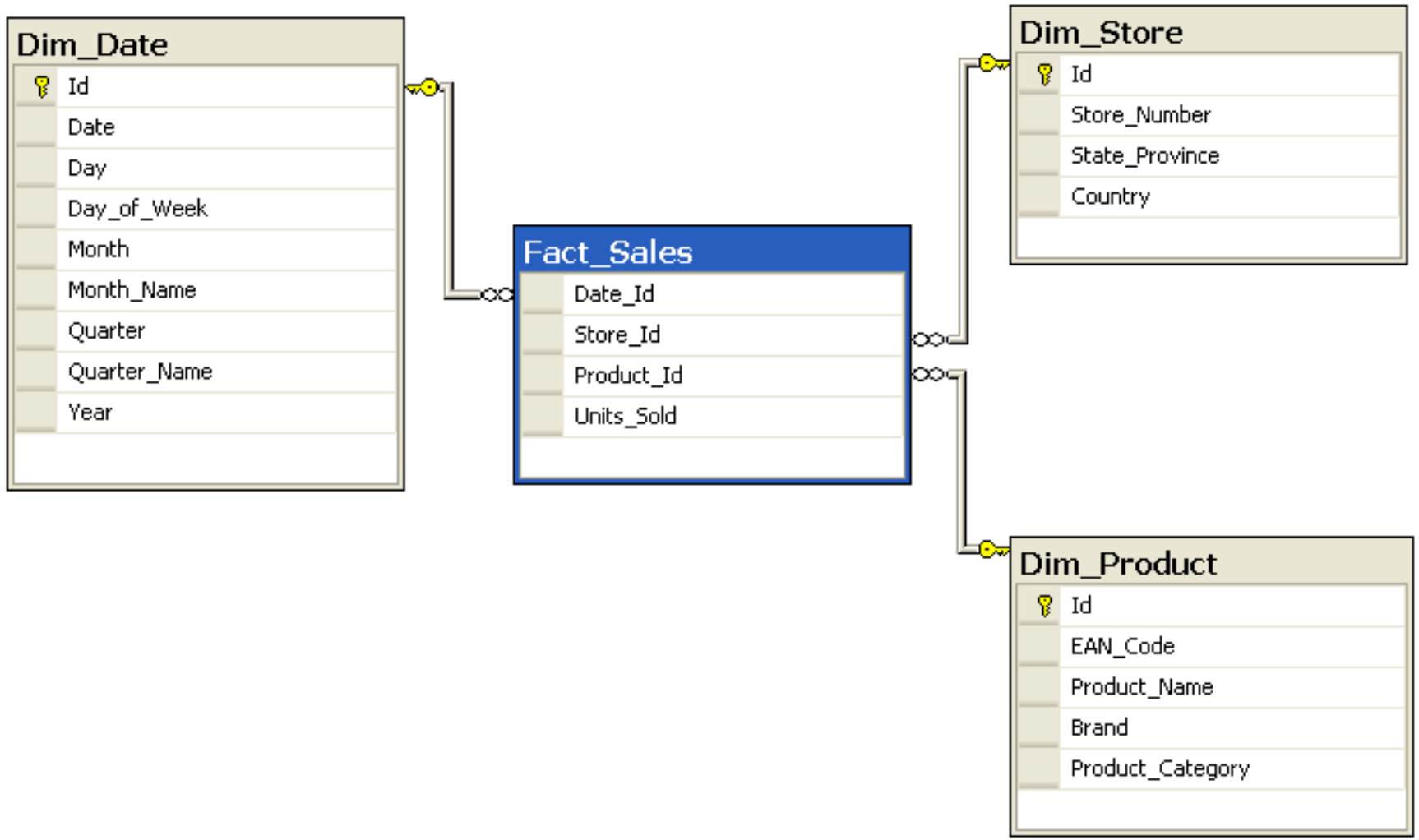
Join statements

- This is what allows us to use “relational” databases

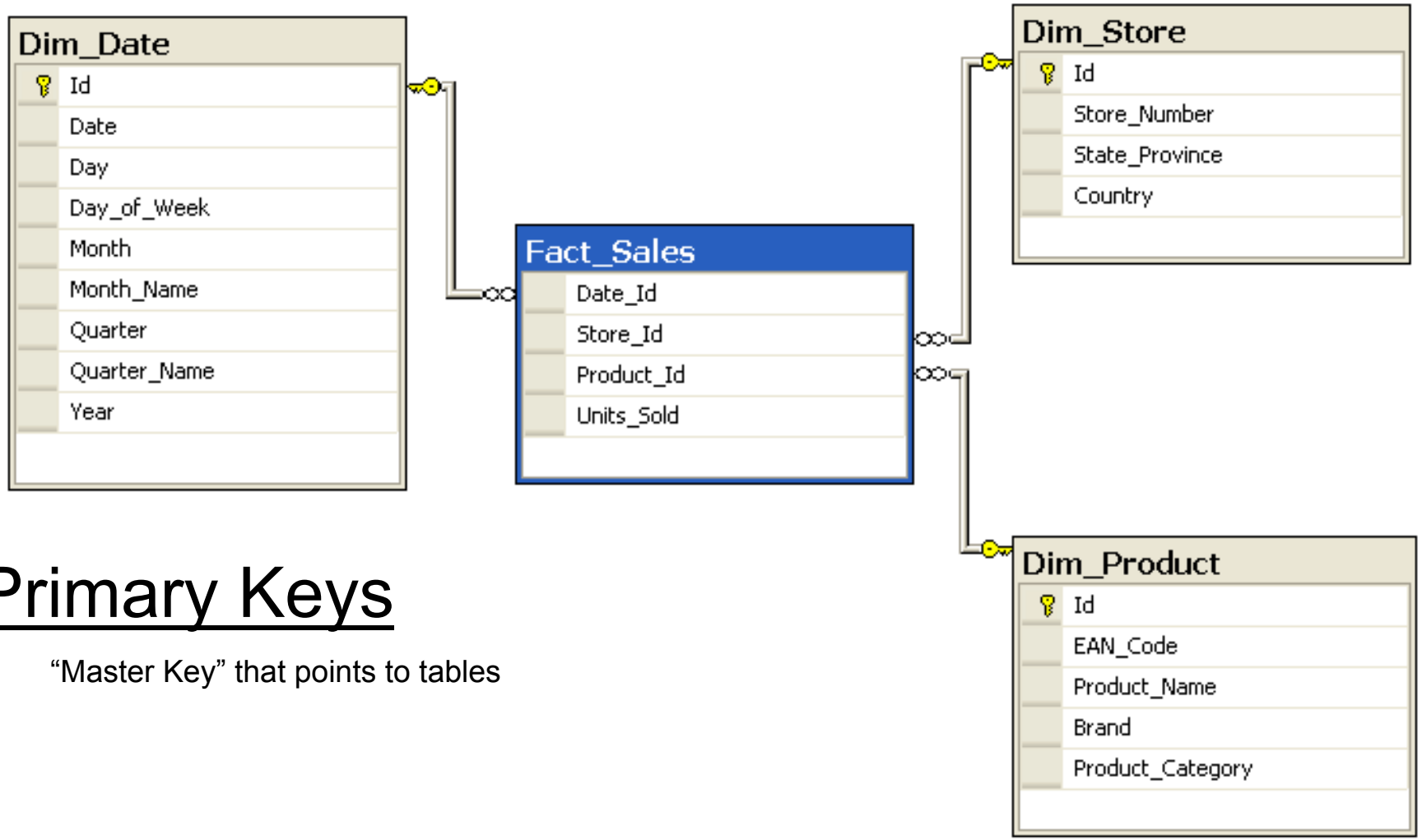
Relational Databases

- Databases with many tables
- Each table “connects” with another table (usually with an Identifying Key or Id)


Tube
tube_assembly_id
material_id
diameter
wall
length
num_bends
bend_radius
end_a_1x
end_a_2x
end_x_1x
end_x_2x
end_a (FK)
end_x (FK)
num_boss
num_bracket
other
Train
tube_assembly_id (FK)
supplier
quote_date
annual_usage
min_order_quantity
bracket_pricing
quantity
cost
Specs
tube_assembly_id (FK)
spec1
spec2
spec3
spec4
spec5
spec6
spec7
spec8
spec9
spec10
Bill_of_Materials
tube_assembly_id (FK)
component_id_1 (FK)
quantity_1
component_id_2 (FK)
quantity_2
component_id_3 (FK)
quantity_3
component_id_4 (FK)
quantity_4
component_id_5
quantity_5
component_id_6
quantity_6
component_id_7
quantity_7
component_id_8
quantity_8
Tube_End_Form
end_form_id
forming
Components
component_id
component_type_id (FK)
name
Type_Component
component_type_id
name
Type_Connection
connection_type_id
name
Type_End_Form
end_form_id
name
Comp_[type]
component_id (FK)
component_type_id (FK)
connection_type_id_1 (FK)
end_form_id_1 (FK)
(65 different attributes)



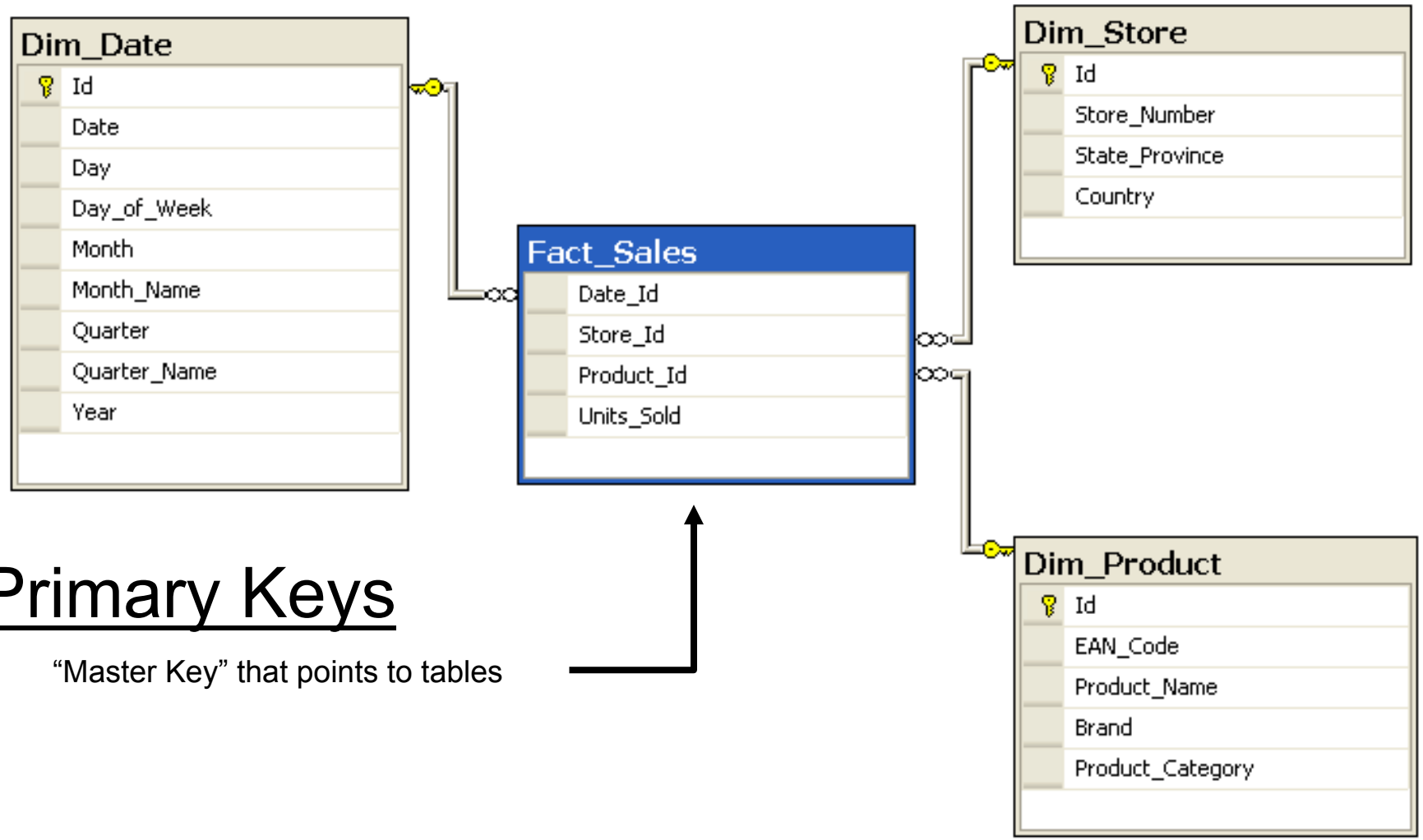
-"Star-schema-example" by SqlPac (talk) - I created this work entirely by myself.. Licensed under CC BY-SA 3.0 via Wikipedia - https://en.wikipedia.org/wiki/File:Star-schema-example.png#/media/File:Star-schema-example.png



Primary Keys

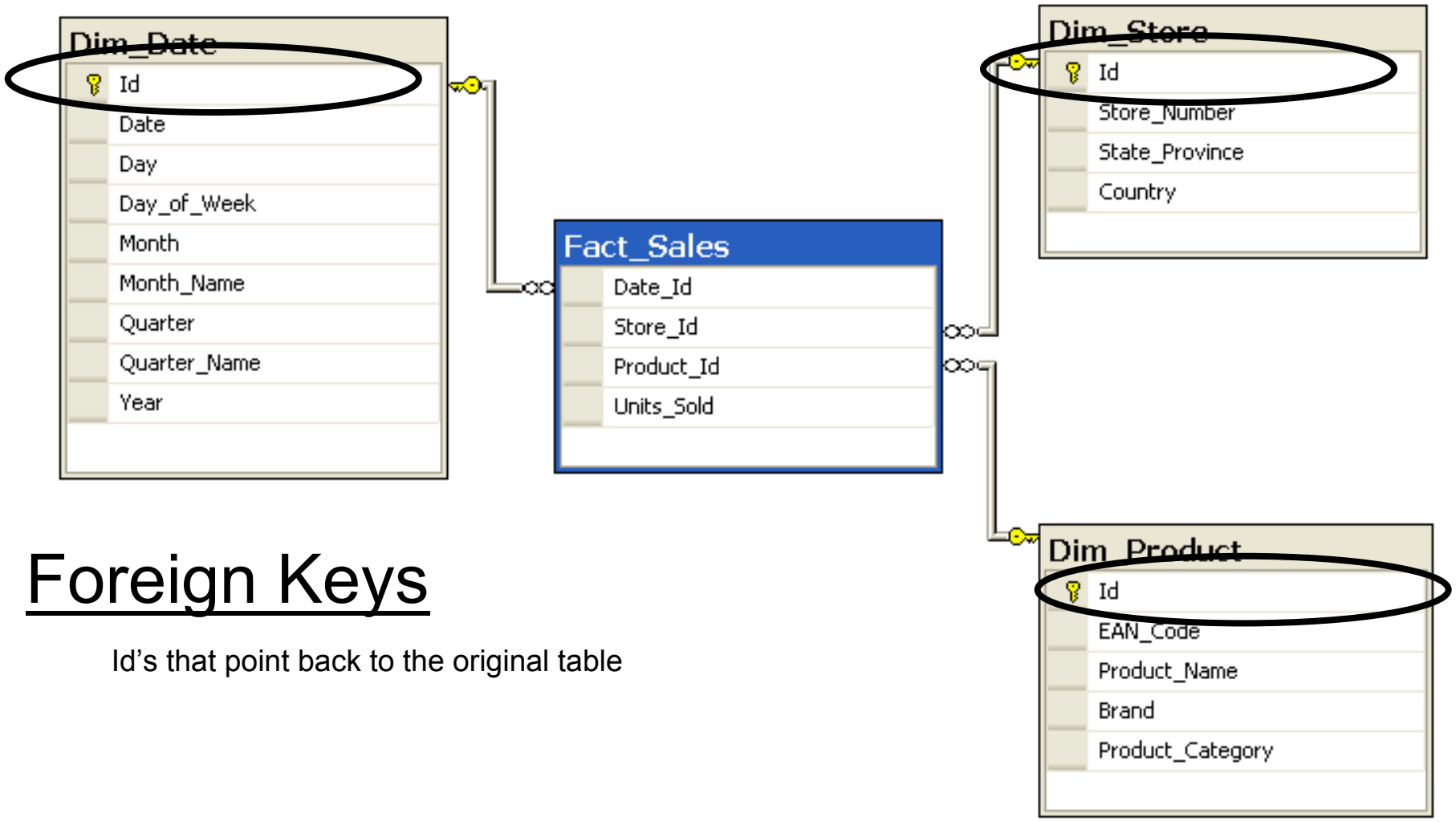
"Master Key" that points to tables

-"Star-schema-example" by SqlPac (talk) - I created this work entirely by myself.. Licensed under CC BY-SA 3.0 via Wikipedia - https://en.wikipedia.org/wiki/File:Star-schema-example.png#/media/File:Star-schema-example.png



Primary Keys

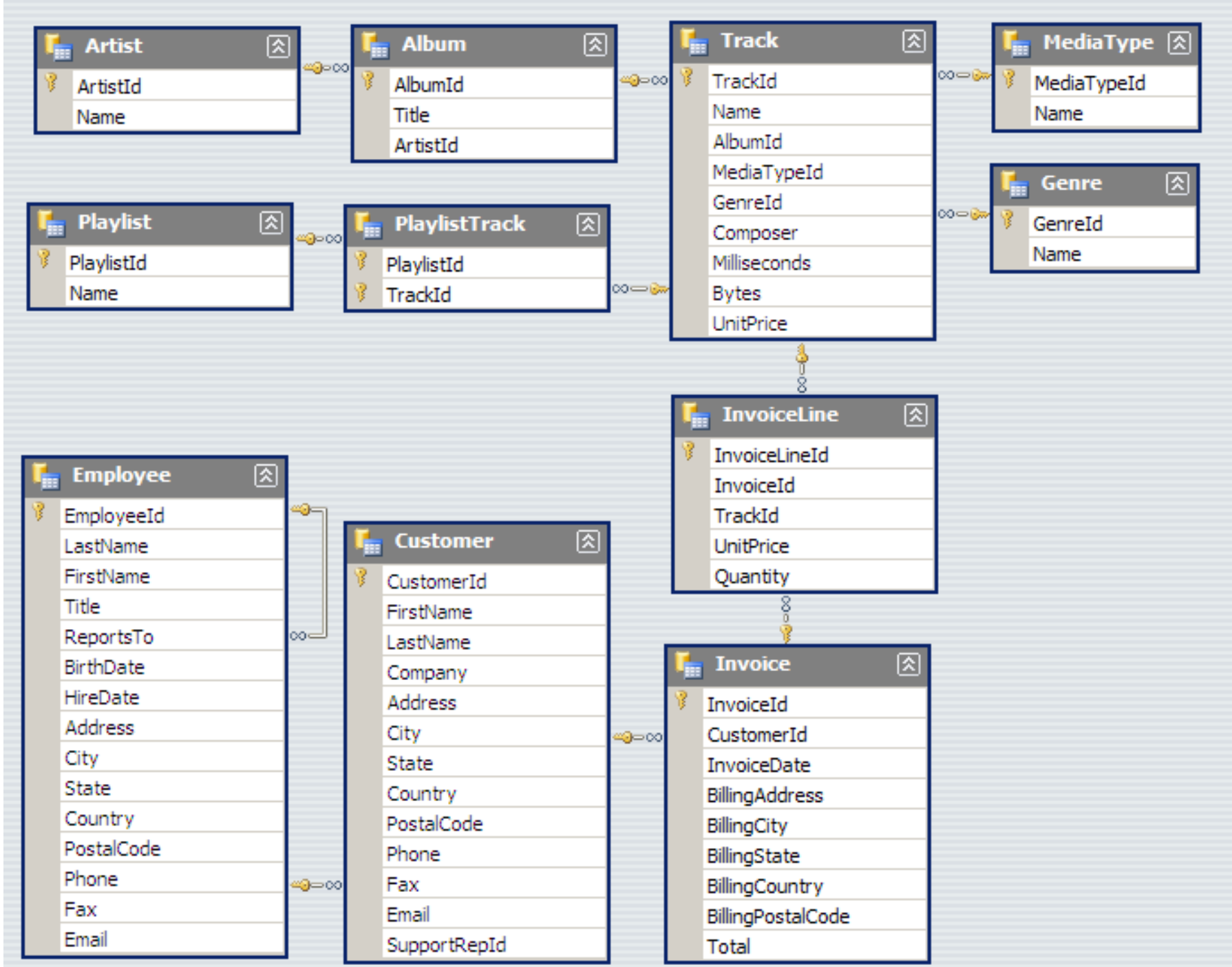
"Master Key" that points to tables



Foreign Keys

Id's that point back to the original table

-"Star-schema-example" by SqlPac (talk) - I created this work entirely by myself.. Licensed under CC BY-SA 3.0 via Wikipedia - https://en.wikipedia.org/wiki/File:Star-schema-example.png#/media/File:Star-schema-example.png


Artist
ArtistId
Name
Album
AlbumId
Title
ArtistId
Track
TrackId
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice
MediaType
MediaTypeId
Name
Genre
GenreId
Name
Playlist
PlaylistId
Name
PlaylistTrack
PlaylistId
TrackId
InvoiceLine
InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity
Employee
EmployeeId
LastName
FirstName
Title
ReportsTo
BirthDate
HireDate
Address
City
State
Country
PostalCode
Phone
Fax
Email
Customer
CustomerId
FirstName
LastName
Company
Address
City
State
Country
PostalCode
Phone
Fax
Email
SupportRepId
Invoice
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total

Joins

Dhruv Relwani
Data Science Society Berkeley
Fall 2016

NULL Values

- Column values are sometimes unknown or inapplicable
  - SQL provides a special value “null”, for such situations.
  - We need a 3-valued logic: True, False and Null (Unknown).

Joins

```
SELECT column1, column2, …

FROM  table1   INNER
               LEFT OUTER
               RIGHT OUTER      JOIN table2
               FULL OUTER

ON table1.PrimaryKey = table2.ForeignKey;
```

Our Tables / Relations

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
95	Bob	3	63.5

Boats

bid	name	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Reserves

sid	bid	day
22	101	10/10/96
95	103	11/12/96

Inner Joins

```
SELECT s.sid, s.sname, r.bi
FROM Sailors s, Reserves r
WHERE s.sid = r.sid;
```

Both are equivalent!

```
SELECT s.sid, s.sname, r.bid
FROM Sailors s INNER JOIN Reserves r
ON s.sid = r.sid;
```

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
95	Bob	3	63.5

Reserves

sid	bid	day
22	101	10/10/96
95	103	11/12/96

s.sid	s.sname	r.bid
22	Dustin	101
95	Bob	103

Left OUTER Joins

- Returns all matched rows from both tables, plus all unmatched rows from the table on the **left** of the "LEFT OUTER JOIN" clause.
  - (use Null where columns don't match)
- Returns all sid, sname for sailors & bid for boats in any of their reservations
  - Note: no match for **s**.sid = **r**.sid? **r**.bid IS NULL!

```
SELECT s.sid, s.sname, r.bid
FROM Sailors s LEFT OUTER JOIN
                Reserves r
ON s.sid = r.sid;
```

s.sid	s.name	r.bid
22	Dustin	101
95	Bob	103
31	Lubber	

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
95	Bob	3	63.5

Reserves

sid	bid	day
22	101	10/10/96
95	103	11/12/96

Right OUTER Joins

- Returns all matched rows, plus all unmatched rows from the table on the **right** of the "RIGHT OUTER JOIN" clause.
  - (use Null where columns don't match)
- Returns all sid for sailors, bid & bname for boats that are reserved.
  - Note: no match for **r**.bid = **b**.bid? **r**.sid IS NULL!

SELECT **r**.sid, **b**.bid, **b**.bname
FROM Reserves **r** RIGHT OUTER JOIN
Boats **b**
ON **r**.bid = **b**.bid;

r.sid	b.bid	b.name
22	101	Interlake
	102	Interlake
95	103	Clipper
	104	Marine

Reserves

sid	bid	day
22	101	10/10/96
95	103	11/12/96

Boats

bid	name	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Full Outer Joins

- Returns all (matched or unmatched) rows from the tables on **both** sides of the "FULL OUTER JOIN" clause.
  - (use Null where columns don't match)
- Returns all sid for sailors, bid & bname for boats that are reserved.
  - Note: no match for **r**.bid = **b**.bid? **r**.sid IS NULL (OR) **b**.bid IS NULL & **b**.bname is NULL

SELECT **r**.sid, **b**.bid, **b**.bname
FROM Reserves **r** FULL OUTER JOIN
Boats **b**
ON **r**.bid = **b**.bid;

r.sid	b.bid	b.name
22	101	Interlake
	102	Interlake
95	103	Clipper
	104	Marine

Reserves

sid	bid	day
22	101	10/10/96
95	103	11/12/96

Boats

bid	name	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

What next?

Practice practice practice!
You can upload *any* sqlite database into sqlitestudio and practice your queries.

Where can I find more data?

1. https://www.kaggle.com/datasets
    a) Great example: https://www.kaggle.com/kaggle/hillary-clinton-emails

Where can I find more functions?

1. http://www.w3schools.com/sql/