Rotman

INTRO TO SQL

BTA SQL Workshop (https://tdmdal.github.io/sql-bta-2021/)

October 20, 2021 Prepared by Jay / TDMDAL

Rotman School of Management
UNIVERSITY OF TORONTO

Goal for Today (2 hrs)

- Understand what’s SQL and related concepts such as
  - Database (DB) and relational DB (RDB)
  - DB management system (DBMS) and RDBMS
- Get a taste of SQL coding
  - Simple column and row operations
  - Simple aggregation
  - simple join operation
- Know what to learn next and where to find free learning resources

What’s SQL (Structured Query Language)

- Most widely used database (DB) language
  - a domain specific language: managing data stored in relational DBs

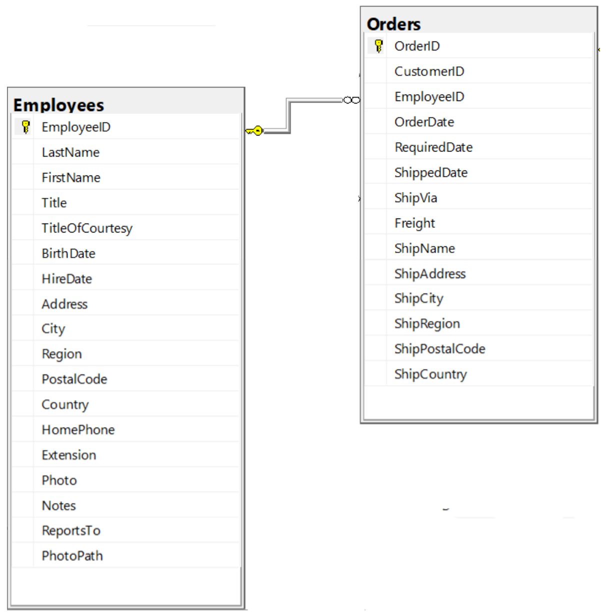
- Not a proprietary language
  - Open specifications/standards (ANSI & ISO)
  - All major DBMS (DB Mgmt. System) vendors implement Standard SQL
  - However, SQL Extensions are usually DB specific (SQL dialects)

- Powerful despite simplicity

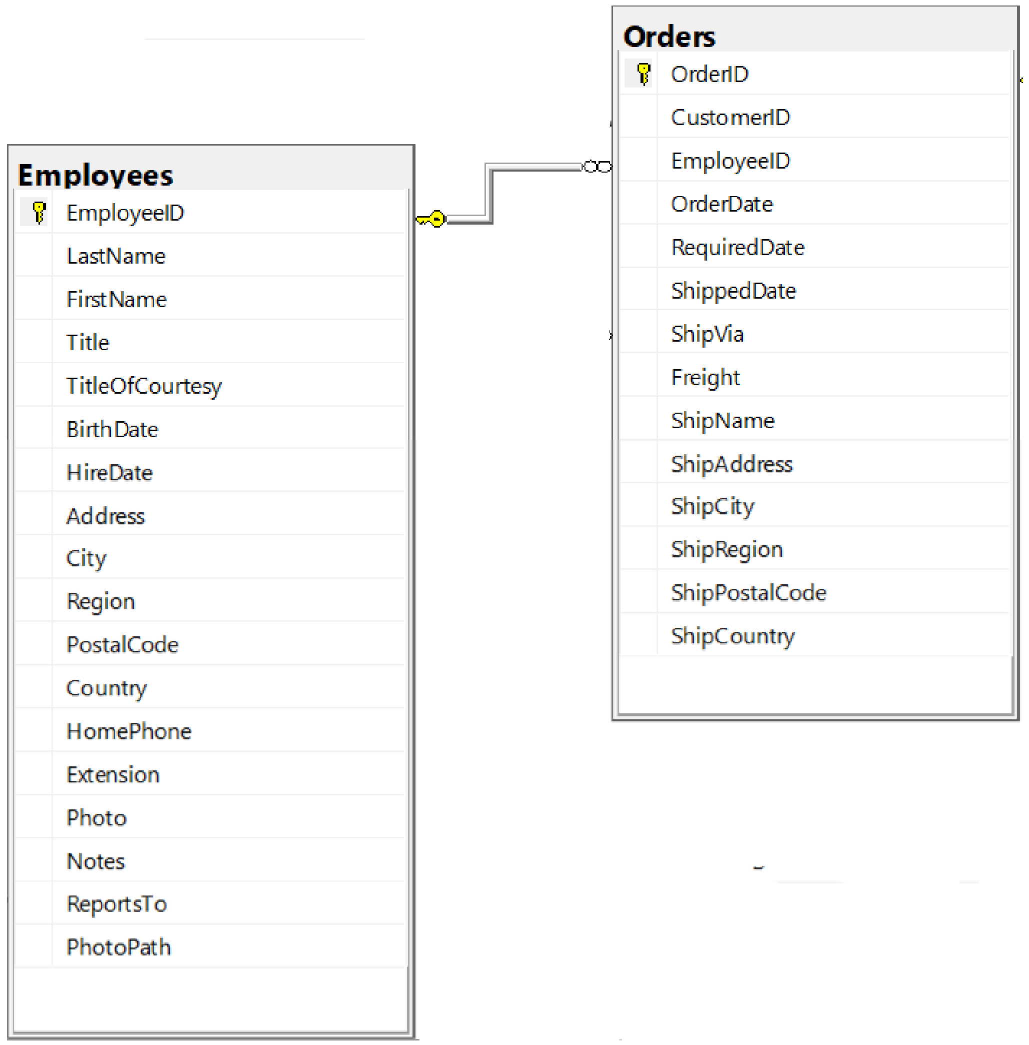
ANSI - American National Standards Institute; ISO – International Organization for Standardization

What’s a DB and a Relational DB

- What’s a database: A collection of data in an organized way
- *Relational DB (RDB)*
  - tables
    - columns/fields/variables and datatypes
    - rows/records/observations
  - primary key, foreign key, constraints and relationships
  - other objects: indices, views, triggers and many more



Primary key (PK), foreign key (FK), constraints & relationships - 1


Orders
- OrderID
- CustomerID
- EmployeeID
- OrderDate
- RequiredDate
- ShippedDate
- ShipVia
- Freight
- ShipName
- ShipAddress
- ShipCity
- ShipRegion
- ShipPostalCode
- ShipCountry

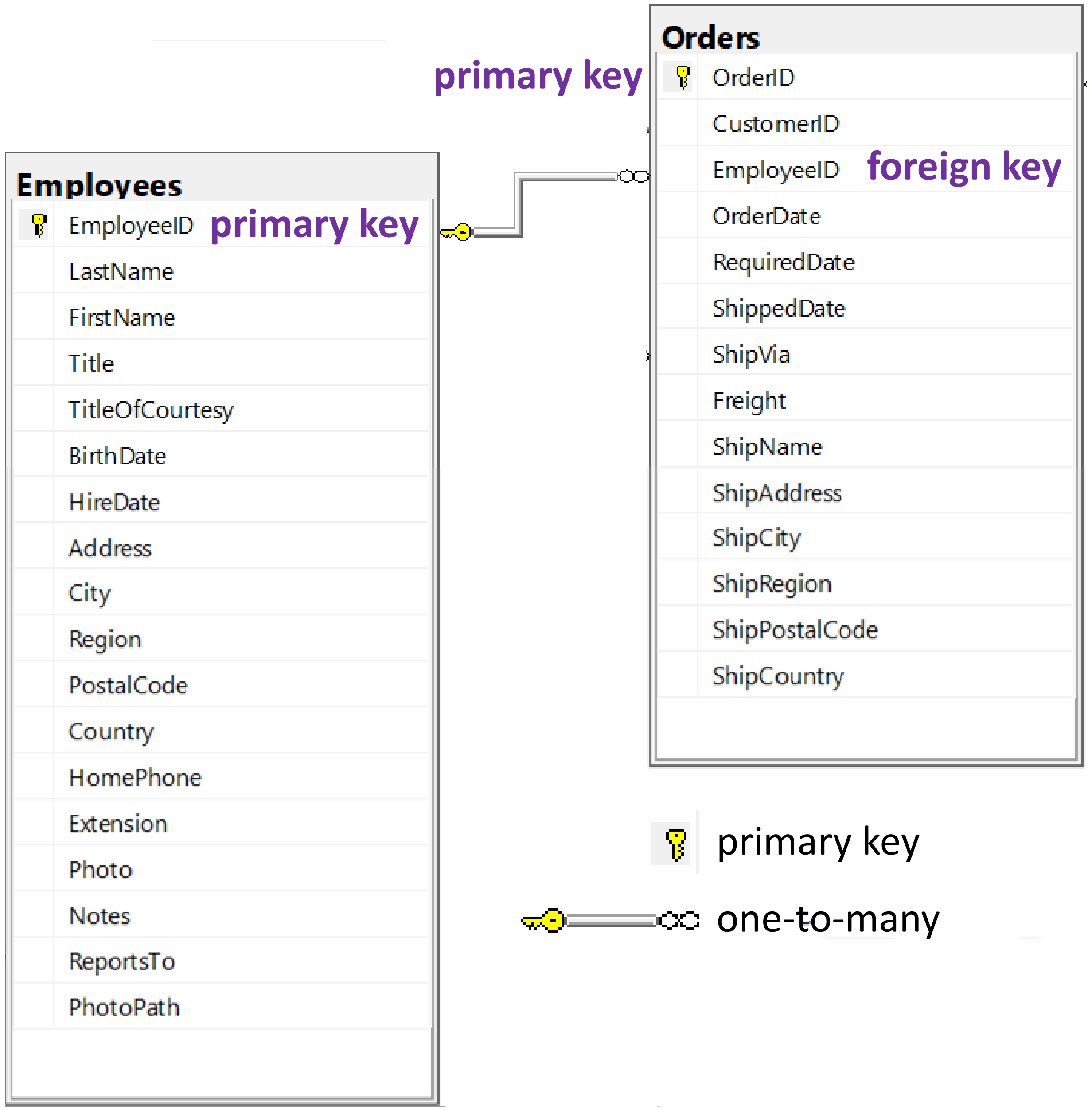
Employees
- EmployeeID
- LastName
- FirstName
- Title
- TitleOfCourtesy
- BirthDate
- HireDate
- Address
- City
- Region
- PostalCode
- Country
- HomePhone
- Extension
- Photo
- Notes
- ReportsTo
- PhotoPath

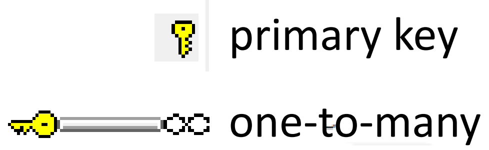
EmployeeID	LastName	FirstName	Title	...
1	Davolio	Nancy	Sales Representative	...
2	Fuller	Andrew	Vice President, Sales	...
3	Leverling	Janet	Sales Representative	...
4	Peacock	Margaret	Sales Representative	...
...	...	...	...	...

OrderID	CustomerID	EmployeeID	...
10248	VINET	5	...
10249	TOMSP	6	...
10250	HANAR	4	...
...	...	...	...

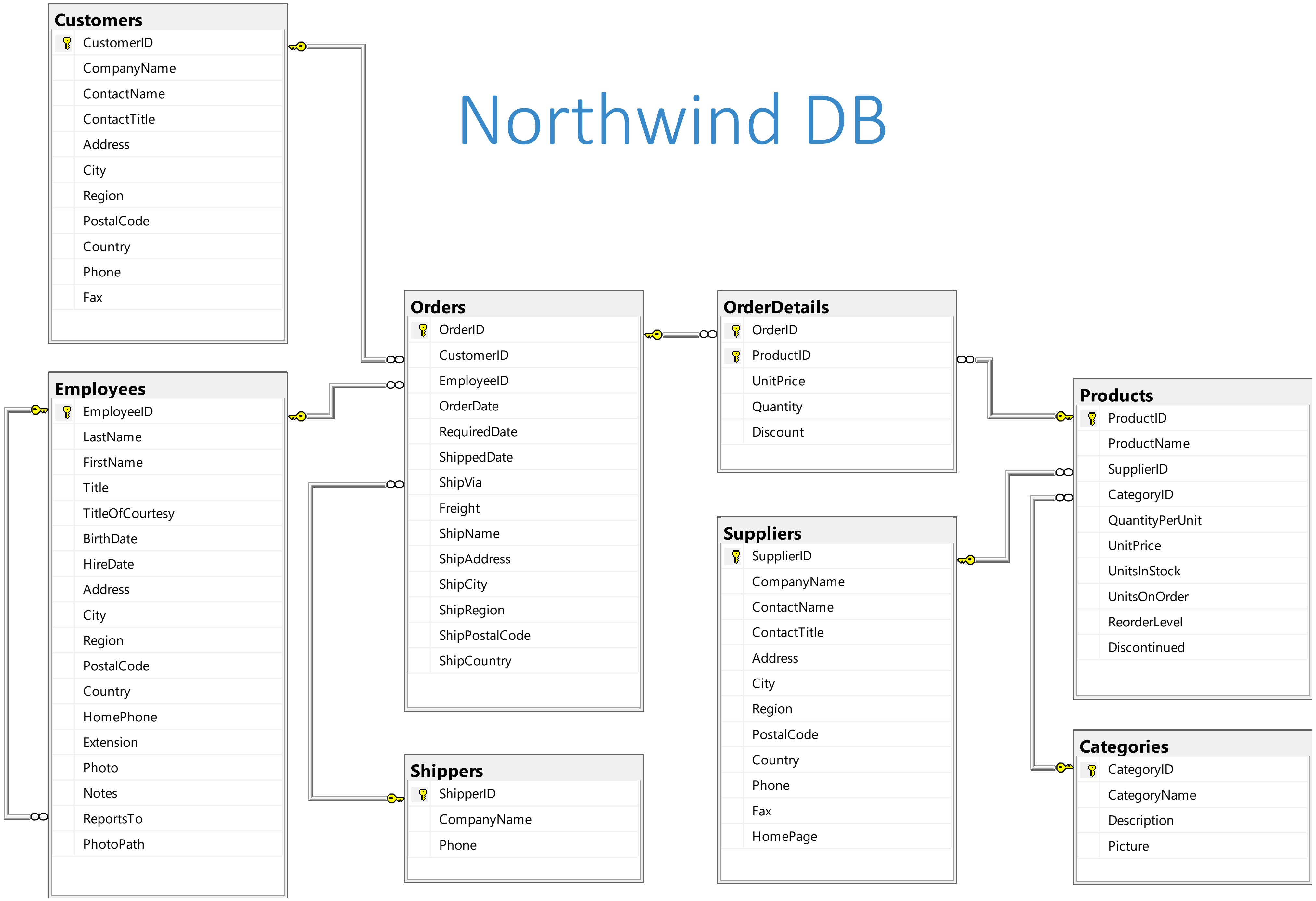
PK, FK, constraints & relationships - 2

- Two keys
  - **primary key**: uniquely identifies an observation in its own table
  - **foreign key**: uniquely identifies an observation in another table
- Relationship between tables
  - one-to-one
  - **one-to-many**
  - many-to-many
- FK constraints





Northwind DB

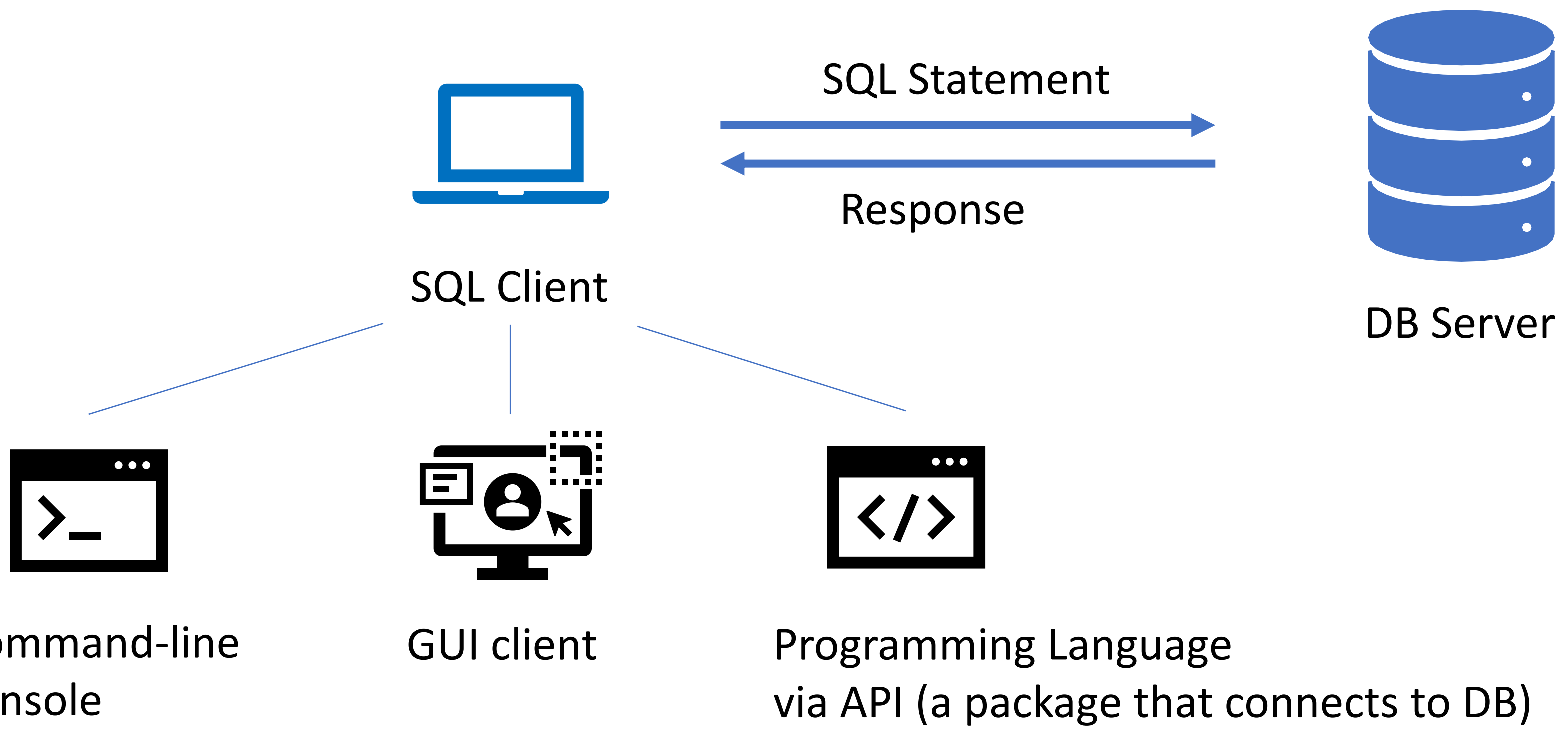


What is a DB Management System

- A software system that manages/maintains DBs

- A few examples of Relational DBMS (RDBMS)
  - Open source: SQLite, MariaDB, PostgreSQL
  - Commercial: MySQL, Microsoft SQL Server, Oracle, etc.

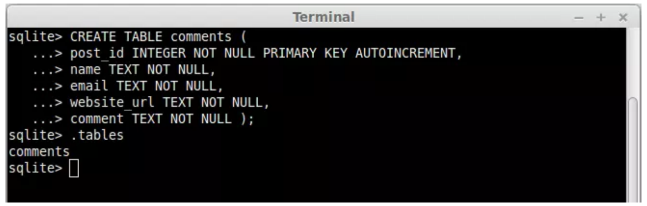
Connect to a DB and write SQL – Architecture



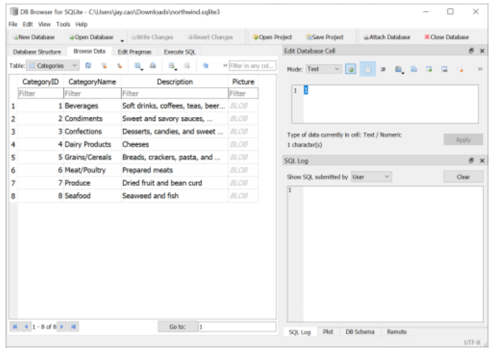
Note: SQL client and DB server can be on the same computer

Connect to a DB and use SQL – SQL Clients

- DB specific management client
  - command-line console
  - GUI (Graphic User Interface) client
    - e.g., DB Browser for SQLite, MySQL Workbench, pgAdmin for PostgreSQL, MS SSMS



```
sqlite> CREATE TABLE comments (
   ...> post_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
   ...> name TEXT NOT NULL,
   ...> email TEXT NOT NULL,
   ...> website_url TEXT NOT NULL,
   ...> comment TEXT NOT NULL );
sqlite> .tables
comments
sqlite> 
```



	CategoryID	CategoryName	Description	Picture
	Filter	Filter	Filter	Filter
1	1	Beverages	Soft drinks, coffees, teas, beer...	BLOB
2	2	Condiments	Sweet and savory sauces, ...	BLOB
3	3	Confections	Desserts, candies, and sweet ...	BLOB
4	4	Dairy Products	Cheeses	BLOB
5	5	Grains/Cereals	Breads, crackers, pasta, and ...	BLOB
6	6	Meat/Poultry	Prepared meats	BLOB
7	7	Produce	Dried fruit and bean curd	BLOB
8	8	Seafood	Seaweed and fish	BLOB

Type of data currently in cell: Text / Numeric
1 character(s)

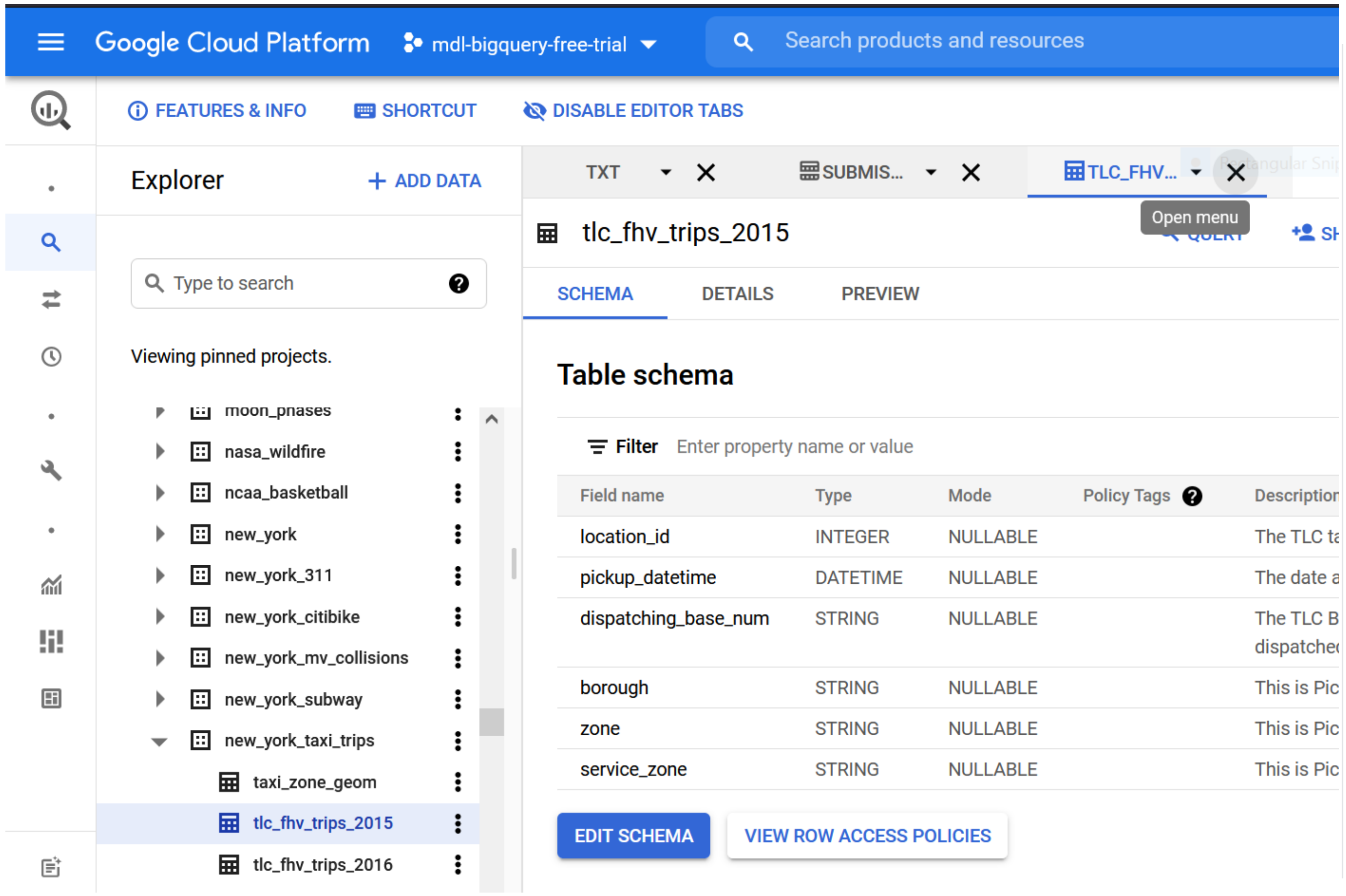
- Generic DB client can connect to different DBs through connectors
  - GUI client (e.g. DBeaver, Beekeeper Studio, Navicat)
  - Programming language
    - e.g., Python + SQLAlchemy + DBAPI (e.g. SQLite, MySQL, PostgreSQL, etc.), R + dbplyr
    - **In this workshop**: Python + ipython-sql notebook magic (depends on SQLAlchemy)

Beyond a relational DB language

- SAS's PROC SQL
- Spark's SparkSQL
  - Apache Spark is a big data computing framework
- Hive's HiveQL, an SQL-like query language
  - Apache Hive is a distributed data warehouse (data warehouse?)
- **Google BigQuery's SQL** (a great first step to big data analysis)
  - BigQuery is Google's data warehouse (analyze petabytes of data at ease)

ref. 1) Database vs data warehouse; 2) Data warehouse vs data lake; 3) NoSQL DB vs SQL DB

Big Data ML with SQL (e.g. Google BigQuery)



tlc_fhv_trips_2015

Table schema

Field name	Type	Mode	Policy Tags	Description
location_id	INTEGER	NULLABLE		The TLC ta
pickup_datetime	DATETIME	NULLABLE		The date a
dispatching_base_num	STRING	NULLABLE		The TLC B dispatche
borough	STRING	NULLABLE		This is Pic
zone	STRING	NULLABLE		This is Pic
service_zone	STRING	NULLABLE		This is Pic

```
CREATE OR REPLACE MODEL taxi.taxifare_model_2
OPTIONS
  (model_type='linear_reg', labels=['total_fare']) AS
WITH params AS (
    SELECT
    1 AS TRAIN,
    2 AS EVAL
    ),
  daynames AS
    (SELECT ['Sun', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri',
'Sat'] AS daysofweek),
  taxitrips AS (
  SELECT
    (tolls_amount + fare_amount) AS total_fare,
    daysofweek[ORDINAL(EXTRACT(DAYOFWEEK FROM
pickup_datetime))] AS dayofweek,
    EXTRACT(HOUR FROM pickup_datetime) AS hourofday,
    SQRT(POW((pickup_longitude - dropoff_longitude),2) +
POW(( pickup_latitude - dropoff_latitude), 2)) as dist,
#Euclidean distance between pickup and drop off
    SQRT(POW((pickup_longitude - dropoff_longitude),2))
as longitude, #Euclidean distance between pickup and drop
off in longitude
    SQRT(POW((pickup_latitude - dropoff_latitude), 2)) as
…
```

Ref. Using BigQuery ML and BigQuery GIS together to predict NYC taxi trip cost

SQL Hands-on Learning (Learning-by-doing)

- Course website: https://tdmdal.github.io/sql-bta-2021/

- Google Colab
  - Google's Jupyter Notebook
  - A notebook can contain live code, equations, visualizations and narrative text

- Why SQLite?
  - a small, fast, self-contained, high-reliability, full-featured, SQL DB engine
  - perfect for learning SQL

Hands-on Part 1: Basics

- Retrieve data: `SELECT...FROM...`
- Filter data: `SELECT...FROM...WHERE...`
  - `IN`, `NOT`, `LIKE` and % wildcard
- Sort retrieved data: `SELECT...FROM...ORDER BY...`
- Create calculated fields
  - mathematical calculations (e.g. `+, -, *, /`)
  - data manipulation functions (e.g. `DATE()`, ||)

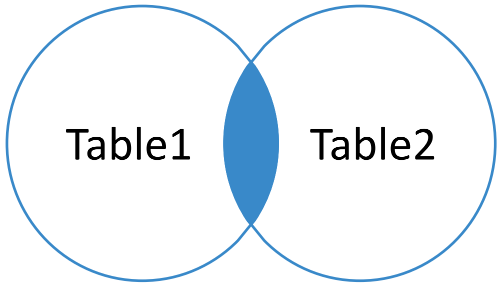
Hands-on Part 2: Summarize and Group Data

- Summarize data using aggregate functions (e.g. `COUNT()`, `MIN()`, `MAX()`, and `AVG()`).
- Group data and filter groups: `SELECT...FROM...GROUP BY...HAVING...`
- SELECT statement syntax ordering
  - `SELECT...FROM...WHERE...GROUP BY...HAVING...ORDER BY...`

Hands-on Part 3: Join Tables

- Inner join: `SELECT...FROM...INNER JOIN...ON...`
- Left join: `SELECT...FROM...LEFT JOIN...ON...`
- Other join variations (see appendix)

Join – Inner Join



```
SELECT *
FROM Table1
  INNER JOIN Table2
  ON Table1.pk = Table2.fk;
```

Table1

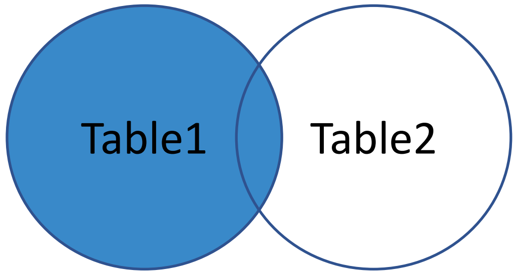
pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

pk	t1c1	fk	t2c1
1	a	1	c
1	a	1	d

Join – Left (Outer) Join



```
SELECT *
FROM Table1
  LEFT JOIN Table2
  ON Table1.pk = Table2.fk;
```

Table1

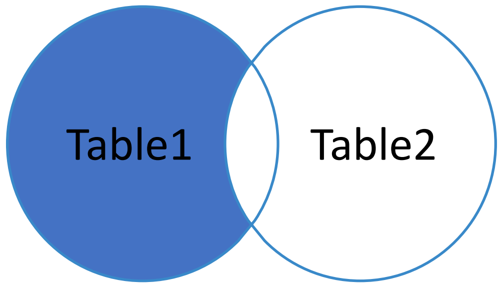
pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

pk	t1c1	fk	t2c1
1	a	1	c
1	a	1	d
2	b	null	null

Join - Left (Outer) Join With Exclusion



```
SELECT *
FROM Table1
  LEFT JOIN Table2
  ON Table1.pk = Table2.fk
WHERE Table2.fk is NULL;
```

Table1

pk	t1c1
1	a
2	b

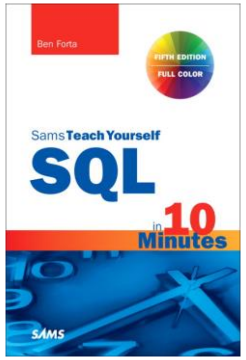
Table2

fk	t2c1
1	c
1	d
3	e

pk	t1c1	fk	t2c1
2	b	null	null

Learning Resources

- A more complete notebook for SQL exercises on workshop website
- Online resources
  - SQLite Tutorial: https://www.sqlitetutorial.net/
  - SQL Tutorial Org: https://www.sqltutorial.org/
  - W3 School SQL tutorial: https://www.w3schools.com/sql/default.asp
  - SQL for Data Analysis at Udacity
  - Learning SQL Programming by Scott Simpson (1h 27m) on LinkedIn Learning
- A little book
  - SQL in 10mins a Day (5th edition) by Ben Forta



Appendix

- Many join operation variations
- SQL is much more…

Join – Right Outer Join*

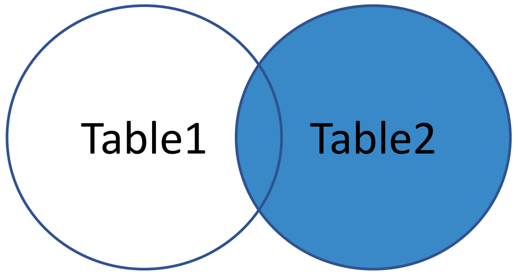


Table1

pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

```
SELECT *
FROM Table2
  LEFT JOIN Table1
  ON Table2.fk = Table1.pk
---------------------------
SELECT *
FROM Table1
  RIGHT JOIN Table2
  ON Table1.pk = Table2.fk;
```

SQLite doesn't support this RIGHT JOIN key word, but some DBMSs do (e.g. MySQL).

pk	t1c1	fk	t2c1
1	a	1	c
1	a	1	d
null	null	3	e

Join - Right Outer Join With Exclusion*

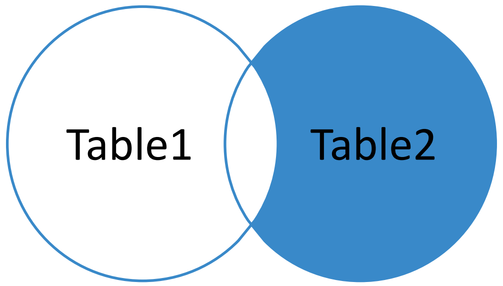


Table1

pk	t1c1
1	a
2	b

Table2

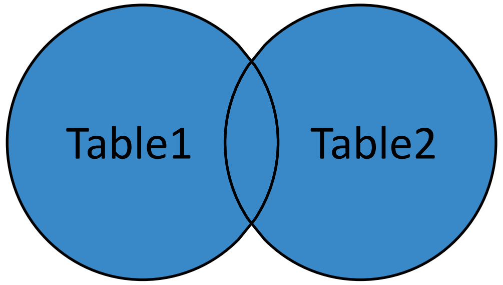
fk	t2c1
1	c
1	d
3	e

```
SELECT *
FROM Table2
  LEFT JOIN Table1
  ON Table2.fk = Table1.pk
WHERE Table1.pk is NULL;
---------------------------
SELECT *
FROM Table1
  RIGHT JOIN Table2
  ON Table1.pk = Table2.fk
WHERE Table1.pk is NULL;
```

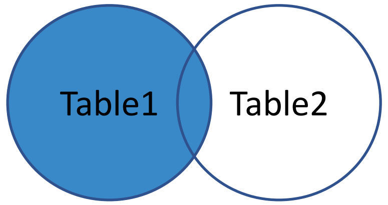
pk	t1c1	fk	t2c1
null	null	3	e

SQLite doesn't support this RIGHT JOIN key word, but some DBMSs do (e.g. MySQL).

Join – Full Outer Join



```
SELECT pk, t1c1, fk, t2c1
FROM Table1
  LEFT JOIN Table2
  ON Table1.pk = Table2.fk
UNION
SELECT pk, t1c1, fk, t2c1
FROM Table2
  LEFT JOIN Table1
  ON Table2.fk = Table1.pk;
```



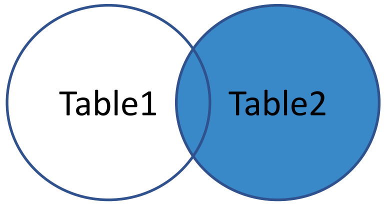


Table1

pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

pk	t1c1	fk	t2c1
1	a	1	c
1	a	1	d
2	b	null	null
null	null	3	e

Note: Some DBMS support FULL OUTER JOIN keyword (e.g. MS SQL) so you don’t need to do it the above way.

Join – Full Outer Join With Exclusion*

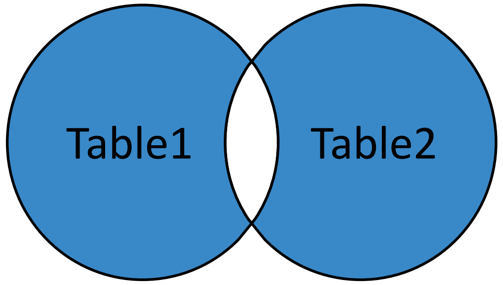


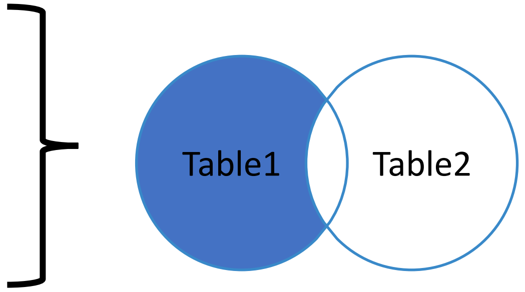
Table1

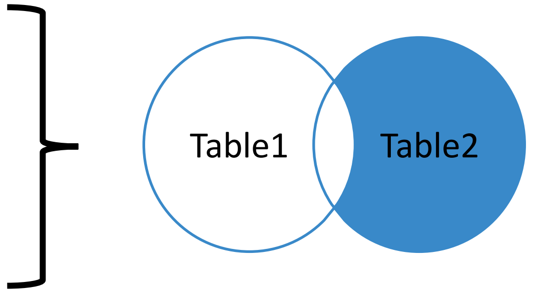
pk	t1c1
1	a
2	b

Table2

fk	t2c1
1	c
1	d
3	e

```
SELECT pk, t1c1, fk, t2c1
FROM Table1
  LEFT JOIN Table2
  ON Table1.pk = Table2.fk
WHERE Table2.fk is NULL
UNION
SELECT pk, t1c1, fk, t2c1
FROM Table2
  LEFT JOIN Table1
  ON Table2.fk = Table1.pk
WHERE Table1.pk is NULL;
```





pk	t1c1	fk	t2c1
2	b	null	null
null	null	3	e

SQL is much more - 1

- Sub-query
- CTE and temporary table
- Self-join
- CASE keyword
- UNION keyword

SQL is much more - 2

- Insert data (`INSERT INTO…VALUES…; INSERT INTO…SELECT…FROM…`)
- Update data (`UPDATE…SET…WHERE…`)
- Delete data (`DELETE FROM…WHERE…`)
- Manipulate tables (`CREATE TABLE…; ALTER TABLE…; DROP TABLE…`)
- Views (`CREATE VIEW…AS…`)

The list goes on and on

- Stored procedures
- Functions
- Transaction processing
- Cursors (going through table row by row)
- WINDOW function
- Query optimization
- DB permissions & security
- …

Ref. A stack overflow discussion on What is “Advanced” SQL.