Advanced SQL Functions

CEE412 / CET522

Transportation Data Management and Visualization

WINTER 2020

Announcement

Corrections

- Drop column from a table (Lecture 5 Slide 53)

```
ALTER TABLE person
 DROP birthdate
```

MySQL Version

```
ALTER TABLE person
 DROP COLUMN birthdate
```

Oracle and SQL Server Versions

Outline

- Window functions
- Case statements
- Variables
- Loops
- Stored procedures

Window Functions

Window functions return a single value for each row based on some operations on the query result.

- Allow more complex sorting and ordering of data including the elusive quantile functions
- These are specific to SQL Server, this does not work the same on all platforms

Basic syntax:

```
SELECT ROW_NUMBER() OVER (PARTITION BY <attributes>
                          ORDER BY <attributes>) AS name
  FROM relation
```

Ranking/aggregate function

Window

Window Functions

Ranking functions supported in SQL Server:

`ROW_NUMBER()`: Return the row number of the output set (e.g., {1, 2, 3, 4}).

`RANK()`: Returns the rank of each row, with duplicates for a tie (e.g., {1, 2, 2, 4}).

`DENSE_RANK()`: Returns the rank without gaps (e.g., {1, 2, 2, 3}).

`NTILE()`: Distributes the rows into a specified number of groups.

Did you ever want to know how to:

- Rank rows based on some attribute value?
- Find the row corresponding to the max or min value of some attribute?
- Compute quantiles, rather than the simple aggregation functions?

Window Functions – Example

Elevation

State	Route	Milepost	Elevation
WA	I-5	0	24.42
WA	I-5	0.001853	24.29
WA	I-5	0.003729	24.16
WA	I-90	0	15.52
WA	I-90	0.005632	15.48
WA	I-90	0.011288	13.79

How to calculate grade for each road segment based on elevation of consecutive points on the road?

- For each point, find the next point on the same road.
- Calculate the elevation difference, then divided by milepost difference.
- Repeat the process for each route separately

Window Functions – Example

But for each point on the road, how to find the NEXT point?

- Without a window function, it can be quite difficult and inefficient.
- If *A* is the current point on the road, the next point *B* has the minimum milepost among all points with higher milepost than *A*.

Solution without window function:

Calculate grade

```
SELECT a.*, (b.Elevation-a.Elevation)/(b.Milepost-a.Milepost)/5280 AS Grade
  FROM Elevation AS a LEFT JOIN Elevation AS b
    ON a.Route = b.Route
       AND b.Milepost = (SELECT MIN(c.Milepost)
                           FROM Elevation AS c
                          WHERE c.Route = a.Route
                                AND c.Milepost > a.Milepost)
```

Subquery that finds milepost of the next point

Window Functions – Example

Query result

State	Route	Milepost	Elevation	Grade
WA	I-5	0	24.42	-0.01329
WA	I-5	0.001853	24.29	-0.01312
WA	I-5	0.003729	24.16	NULL
WA	I-90	0	15.52	-0.00135
WA	I-90	0.005632	15.48	-0.05659
WA	I-90	0.011288	13.79	NULL

The result is what I want. But is that a good solution?

- Inefficient and slow
- For each point, I need to look into the entire table to find the NEXT point.

Window Functions – Example

Solution using the window function:

1. Create a new column that shows row numbers of the table, within each road, the rows are ordered by milepost.

```
SELECT *, ROW_NUMBER() OVER (PARTITION BY Route
                             ORDER BY Milepost) AS PointOrder
  INTO #Elev_Ordered
  FROM Elevation
```

State	Route	Milepost	Elevation	PointOrder
WA	I-5	0	24.42	1
WA	I-5	0.001853	24.29	2
WA	I-5	0.003729	24.16	3
WA	I-90	0	15.52	1
WA	I-90	0.005632	15.48	2
WA	I-90	0.011288	13.79	3

Window Functions – Example

2. For each point, find the next point on the same road.
3. Calculate the elevation difference, then divided by milepost difference.

```
SELECT a.*, (b.Elevation-a.Elevation)/(b.Milepost-a.Milepost)/5280 AS Grade
  FROM #Elev_Ordered AS a LEFT JOIN #Elev_Ordered AS b
    ON a.Route = b.Route
       AND b.PointOrder = a.PointOrder + 1
```

B is the next point of A

State	Route	Milepost	Elevation	Grade
WA	I-5	0	24.42	-0.01329
WA	I-5	0.001853	24.29	-0.01312
WA	I-5	0.003729	24.16	NULL
WA	I-90	0	15.52	-0.00135
WA	I-90	0.005632	15.48	-0.05659
WA	I-90	0.011288	13.79	NULL

Window Functions

Quantile function: NTILE(n)

- Equally divide the rows into n groups.
- E.g., NTILE(4) means quartiles, NTILE(5) means quintile, etc.
- This does not give you the value of the cut points, it just assigns each row to a particular quantile group.
- For each row, NTILE(n) function will return the number of the group (1~n) to which the row belongs

Window Functions – Example

Create the income percentile for CEOs:

```
SELECT *, NTILE(100) OVER (ORDER BY OneYrPay DESC) as Percentile
  FROM CEOs
 ORDER BY OneYrPay DESC
```

200 CEOs in total, two in each percentile group

Name	Company	OneYrPay	FiveYrPay	Shares	Age	Percentile
John H Hammergren	McKesson	131.19	285.02	51.9	53	1
Ralph Lauren	Ralph Lauren	66.65	204.06	5010.4	72	1
Michael D Fascitelli	Vornado Realty	64.405	NULL	171.7	55	2
Richard D Kinder	Kinder Morgan	60.94	60.94	8582.3	67	2
David M Cote	Honeywell	55.79	96.11	21.5	59	3
George Paz	Express Scripts	51.525	100.21	47.3	57	3
Jeffery H Boyd	Priceline.com	50.185	90.3	128.2	55	4
…	…	…	…	…	…	…

Window Functions

Common functions that can be used over a window:

- Ranking functions: ROW_NUMBER(), RANK(), DENSE_RANK(), NTILE(n), etc.
- Aggregate function: AVG(), MIN(), MAX(), SUM(), COUNT(), etc.

When using aggregate functions over a window, ORDER BY is not used in the window (as it does not make sense).

Case Statements

CASE statements in SQL are one way to return conditional values in a query.

They can be slow compared to regular set-based operations, but can be very useful in some situations. The basic form of a CASE statement is as follows:

```
SELECT CASE <column name>
          WHEN <condition 1> THEN <value 1>
          WHEN <condition 1> THEN <value 1>
          ...
          ELSE <value x>
       END
```

- To be interpreted as: when the column is <condition 1>, return <value 1>, when the column is <condition 2>, then return <value 2>, …, else, return <value x>.

Case Statements

Example:

```
SELECT Name, Section,
       Case Section
         WHEN 'CEE 412' THEN 50
         WHEN ‘CET 522' THEN 60
       END AS TotalPoints
  FROM Students
```

Students

Name	Section
A	CEE 412
B	CET 522
C	CET 522
D	CEE 412
E	CET 522
F	CEE 412
…	…

Name	Section	TotalPoints
A	CEE 412	50
B	CET 522	60
C	CET 522	60
D	CEE 412	50
E	CET 522	60
F	CEE 412	50
…	…	

Variables in SQL

In SQL, a local variable is an object that can hold a single data value of a specific type.

Syntax to declare a variable:

```
DECLARE @variable_name <data type>
SET @variable_name = <some value>
```

Or:

```
DECLARE @variable_name <data type> = <some value>
```

Variables in SQL – Example

Player (Name, Salary, Height, Weight, Team)

Question: find the name of the player with the highest salary.

- Solution using a subquery:

```
SELECT name, salary
  FROM player
 WHERE salary = (SELECT MAX(salary) FROM player)
```

- Solution using a local variable:

```
DECLARE @max_salary INT = (SELECT MAX(salary) FROM player)

SELECT name, salary
  FROM player
 WHERE salary = @max_salary
```

name	salary
Peyton Manning	15000000.00

Loops in SQL

There are several loop types in SQL, we will look at WHILE loops

SQL is not a regular programing language, most things that appear to be solved by loops can in fact be solved using the SQL “set-based” approach

Do not use a loop when a conventional query will do (slow and resource intensive)

Loops in SQL – Example

Create a table with ten rows of random numbers.

1. Create an empty table and a counting variable:

```
CREATE TABLE #temp(ID INT, RandNum DECIMAL(5,4))
DECLARE @counter INT = 1
```

2. Insert values into the table in a WHILE loop

```
WHILE @counter <= 10
  BEGIN
    INSERT INTO #temp VALUES(@counter, RAND())
    SET @counter = @counter + 1
  END
```

Random number (0~1) generator

Loops with IF/BREAK

Stop the loop when the sum of random numbers exceeds 3

```
CREATE TABLE #temp(ID INT, RandNum DECIMAL(5,4))
DECLARE @counter INT = 1

WHILE @counter <= 10
  BEGIN
    INSERT INTO #temp VALUES(@counter, RAND())
    SET @counter = @counter + 1
    IF (SELECT SUM(RandNum) FROM #temp) > 3.0 BREAK
    ELSE CONTINUE
  END
```

Loop with IF/BREAK

Results from previous two queries:

ID	RandNum
1	0.9953
2	0.8091
3	0.9167
4	0.2714
5	0.1149
6	0.9743
7	0.7772
8	0.8559
9	0.7972
10	0.4414

VS.

ID	RandNum
1	0.2238
2	0.7479
3	0.4861
4	0.9626
5	0.5082
6	0.1083

Stored Procedures

A set of saved commands in SQL that can be simply executed at any time and even input parameter values like a function.

Why?

- To minimize the amount of SQL code in a software application.
- To manage access and isolate the SQL logic from the programing logic.

Why Not?

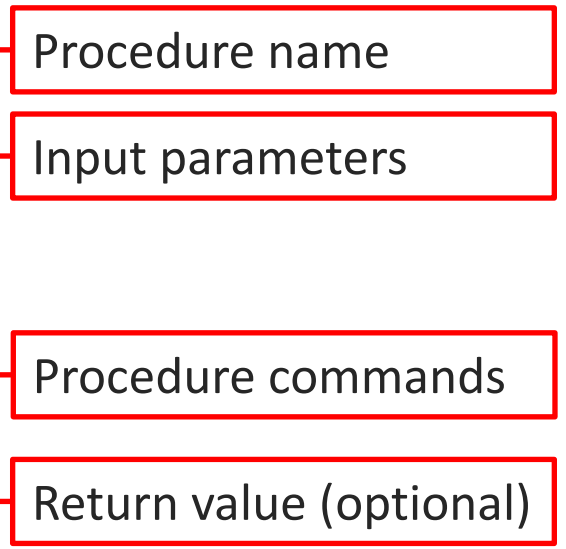
- In current generation SQL Server, little (if any) performance benefits.
- Possibly more work to create and manage procedures in a separate interface.

Stored procedures are fantastic time saving tools for larger or more complex SQL operations. For simple updates and inserts, just use simple queries.

Stored Procedures – Example

Procedure that gives extra credit to students:

```
CREATE PROCEDURE ExtraCredit
  @student_name VARCHAR(10),
  @extra_credit FLOAT
AS
UPDATE #student
   SET Grade = Grade + @extra_credit
 WHERE name = @student_name
RETURN(1)
```



Execute the procedure:

```
EXEC ExtraCredit @student_name = 'Peter', @extra_credit = 5.0
```

Index

When we query data from a table, how can we speed up the query?

- Write a good (simple and accurate) queries
- **Build index for that table**

An index contains keys built from **one or more columns** in the **table** or **view**.

These keys are stored in a tree structure that enables SQL Server to find the row or rows associated with the key values quickly and efficiently.

Index

How to create index?

```
CREATE INDEX index_name
ON table_name (column1, column2, ...)
```

```
DROP INDEX table_name.index_name
```

Create an index on the YearMade column?

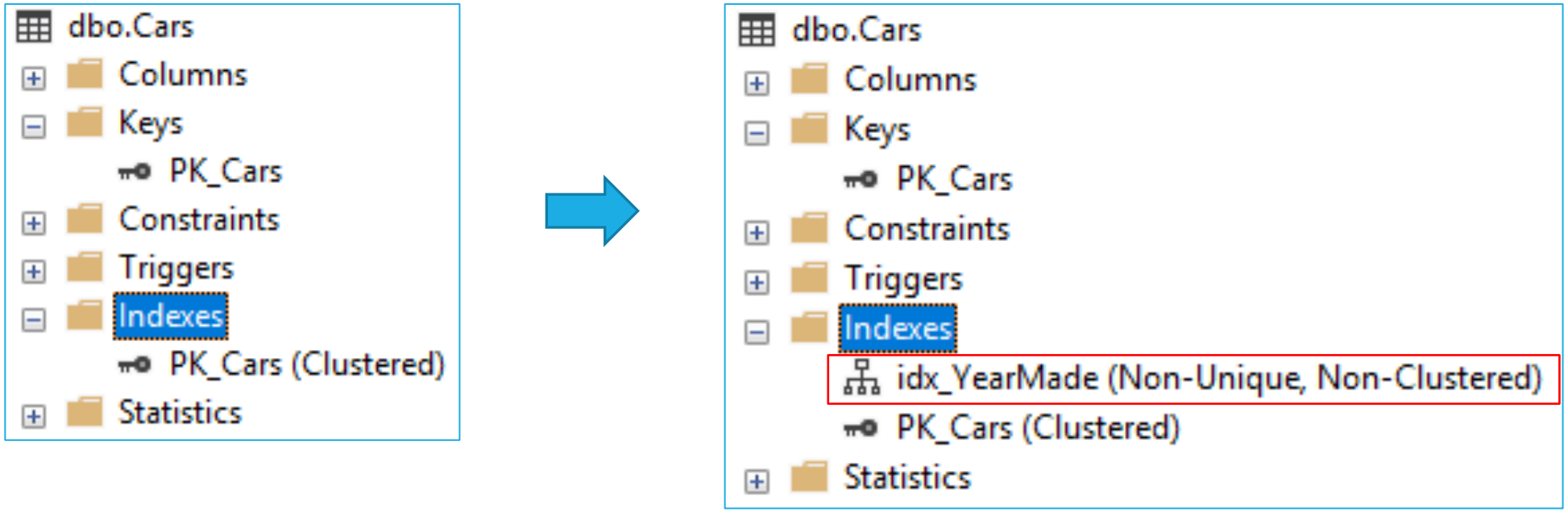
```
CREATE INDEX idx_YearMade
ON  Cars (YearMade)
```

```
DROP INDEX Cars.idx_YearMade
```

LicenseNO	Make	YearMade	OwnerSSN	Type
123ABC	Ford	1990	123-44-5678	Sedan
234BCD	GM	2005	111-22-3333	SUV
345CDE	Toyota	2003	222-33-4444	Sedan
456DEF	Toyota	2004	222-33-4444	Pickup
567XYZ	BMW	1980	120-33-4567	Sedan

Index

Create an Index in SQL Server?



Clustered index?

Non-clustered index?

Index

Clustered

- Clustered indexes sort and store the data rows in the table or view based on their **key values**. These are the columns included in the index definition. There can be **only one clustered index per table**, because the data rows themselves can be stored in only one order.

Non-clustered

- Non-clustered indexes have a structure separate from the data rows. A non-clustered index contains the **non-clustered index key values** and each key value entry has a **pointer to the data row** that contains the key value.

Index

Will the index always speed up Queries?

No…

An index helps to speed up **SELECT** queries and **WHERE** clauses,

but it slows down data input with the **UPDATE** and the **INSERT** statements.

Indexes can be created or dropped with no effect on the data.

Other Databases and Tools for Transportation Data Management

Transportation data

Normally Transportation Data is **spatial-temporal** data

- Loop detector (sensor-based) data
- Incident data
- Weather data
- Traffic counts

In real work, how do we store spatial temporal data?

- Take **loop detector data** as an example:

stamp	loopid	speed	volume	occupancy	goodfrac
1/1/15 0:05	1	65.08282	0	0	1
1/1/15 0:10	1	65.08282	0	0	1
1/1/15 0:15	1	64.31717	0	0	1
1/1/15 0:20	1	64.06195	1	0.000947	1
1/1/15 0:25	1	65.08282	0	0	1
1/1/15 0:30	1	65.08282	0	0	1

Transportation data

How can we link those spatial-temporal data with the specific sensors?

- Store sensor's information in another table (Loop detector's **cabinet table**)

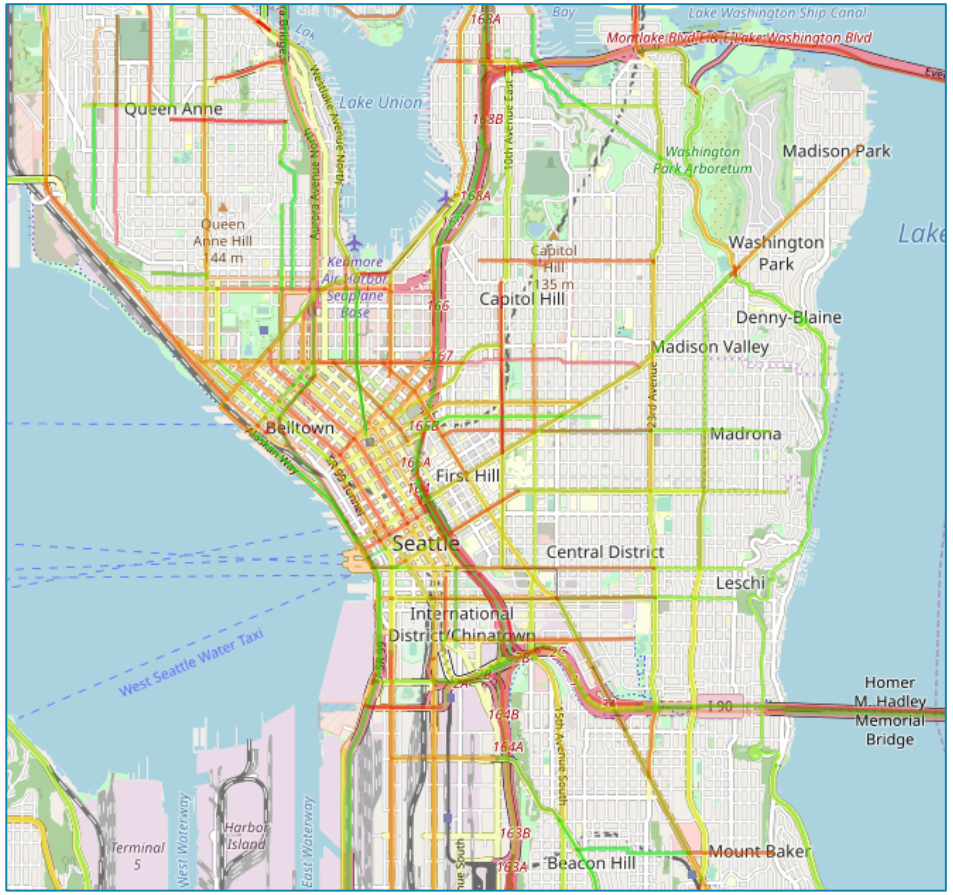
CabName	UnitType	ID	Lat	Lon	Route	Milepost	direction	UnitName
002es00068	main	1	47.97841	-122.177	2	0.68	E	002es00068:_ME___1
002es00068	main	2	47.97841	-122.177	2	0.68	E	002es00068:_ME___2
002es00068	main	3	47.97841	-122.177	2	0.68	E	002es00068:_ME___3
002es00068	speed	19548	47.97841	-122.177	2	0.68	E	002es00068:_ME__S1

The geolocation of sensors can be stored by two columns, i.e. latitude and longitude.

What if a data set measure the traffic states of road segments?

Transportation data

Let’s see INRIX data



How can we store road segment’s geolocation information?

Can we do it in a similar way?

Yes…

Transportation data

Example

- IRNIX TMC Table (TMC: Traffic Message Channel)

TMC	State	County	Road No.	Direction	StartLat	StartLong	EndLat	EndLong	Miles
101-06899	GA	FULTON	US-29	S	33.55658	-84.5932	33.54987	-84.6037	0.762831
101-06917	GA	COBB		W	33.96469	-84.4998	33.96218	-84.5173	1.116469
101-06918	GA	COBB		N	33.94558	-84.4987	33.9572	-84.4965	0.81509
101-06921	GA	COBB		W	33.92285	-84.4921	33.92279	-84.5039	0.682235
101-06928	GA	FLOYD	US-411	S	34.15415	-85.2703	34.10673	-85.352	6.067722

What if the road segments are not straight lines, but curved lines?

Transportation data

We need more powerful datatypes to store those curved lines.

Another database, PostgreSQL, can help us.

PostgreSQL

- A powerful, open source object-relational database system
- Long history
- Well documented
- Comprehensive data types
- Powerful extensions
- Open source

PostgreSQL

SQL Server → MS SQL Server Management Studio

PostgreSQL → pgAdmin

PostgreSQL has a lot of features.

- User-defined types.
- Table inheritance.
- Sophisticated locking mechanism.
- Foreign key referential integrity.
- Views, rules, subquery.
- Nested transactions (savepoints)
- Multi-version concurrency control (MVCC)
- Asynchronous replication.

But Let's come back to our problem: store curved lines?

PostgreSQL

PostgreSQL support geospatial or GIS data.

- Geospatial data is represented by **vectors**, stored in files usually called **shapefile**.
- A bunch of connected vectors/segments → the curved road segments

Example: (INRIX road geometric data table)

The starting and ending points of a straight/curved line

objectid	shape	link_id	st_name	feat_id	dir_travel	iso_code	shape_leng	source	target
1	0105000020E610000001000000010200000002000000004C9B17CFA0955EC098D8CE3637CE4740A8A7CDE49B955EC0581353793BCE4740	19382442	PINE ST	7.41E+08	T	USA	0.000327	1	2
2	0105000020E610000001000000010200000002000000004320908A38B5EC000BED1A92B0F484074A79C11A58B5EC02007BE82340F4840	21016083	SMOKEY POINT BLVD	7.17E+08	B	USA	0.000285	3	4

PostgreSQL

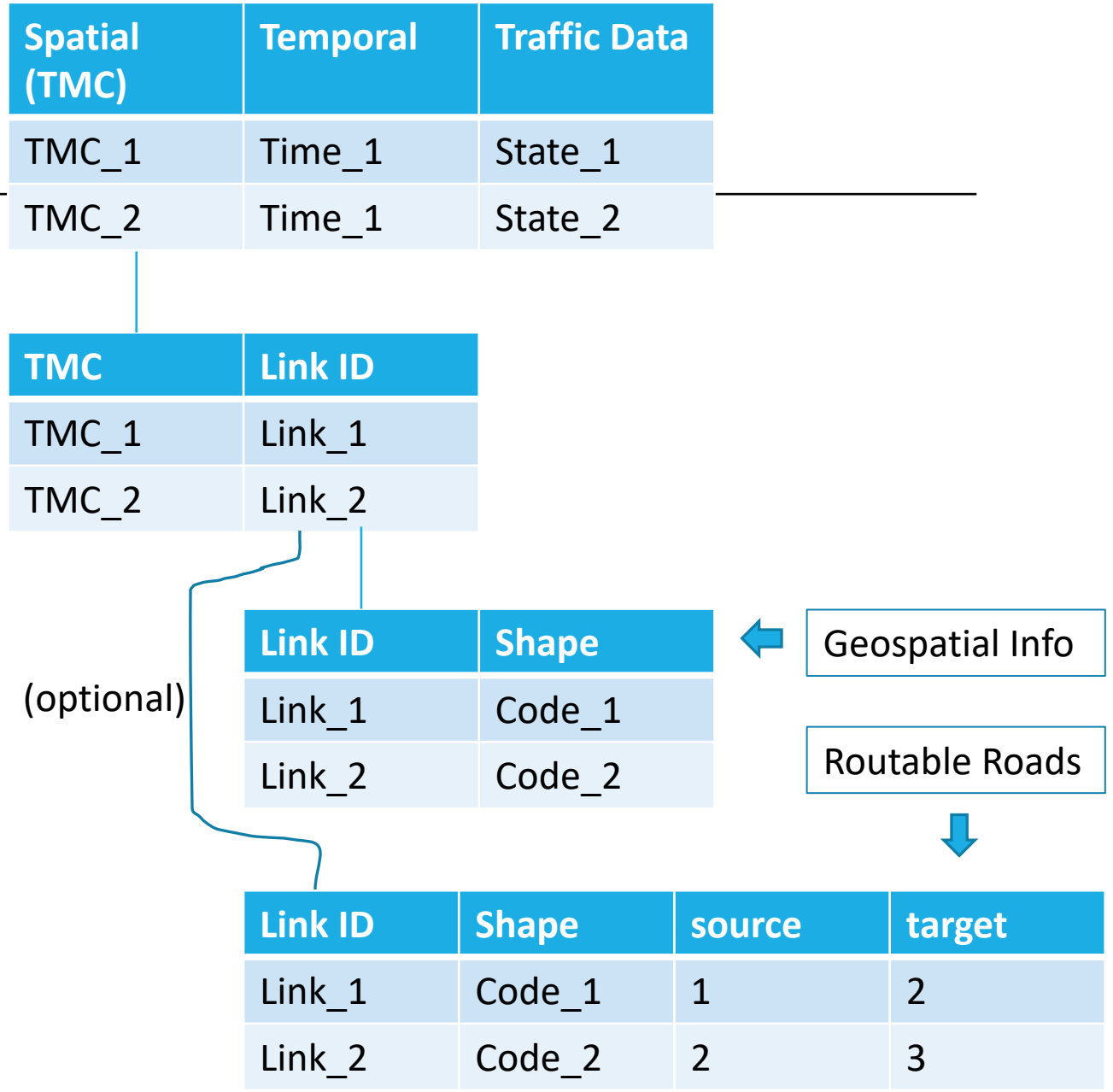
Big picture of transportation data (INRIX data) storage in PostgreSQL

- (Spatial, Temporal) → Traffic data
- Spatial → Road link id
- Road link id → shape (geospatial info)
- Shape of road can be routable

How to make the roads stored in PostgreSQL be routable?

PotsGIS and pgRouting

- https://postgis.net/
- https://pgrouting.org/



PostGIS

PostGIS is a spatial database extender for PostgreSQL

- **Support** many **GIS** functionalities as like finding nearest neighbor, distance calculation from one point to another.
- Example:

```
SELECT superhero.name
FROM city, superhero
WHERE ST_Contains(city.geom, superhero.geom)
AND city.name = 'Gotham';
```

  - geom: geometry can be a point, a line, a polygon, a polygon with a hole, or a collection.
  - ST_Contains(A, B):Geometry A contains Geometry B

- Important to transportation data management
  - Calculating the distance between an incident and a sensor
  - Separating roadway into small segments

pgRouting

pgRouting extends the PostGIS / PostgreSQL to provide geospatial routing functionality.

- Example (you can find more in pgRouting document: http://docs.pgrouting.org/)

```
SELECT * FROM pgr_dijkstra(
    'SELECT id, source, target, cost, reverse_cost FROM edge_table',
    2, 3,
    FALSE
);
 seq | path_seq | node | edge | cost | agg_cost
-----+----------+------+------+------+----------
   1 |        1 |    2 |    2 |    1 |        0
   2 |        2 |    3 |   -1 |    0 |        1
(2 rows)
```

Sequence of Shortest Path

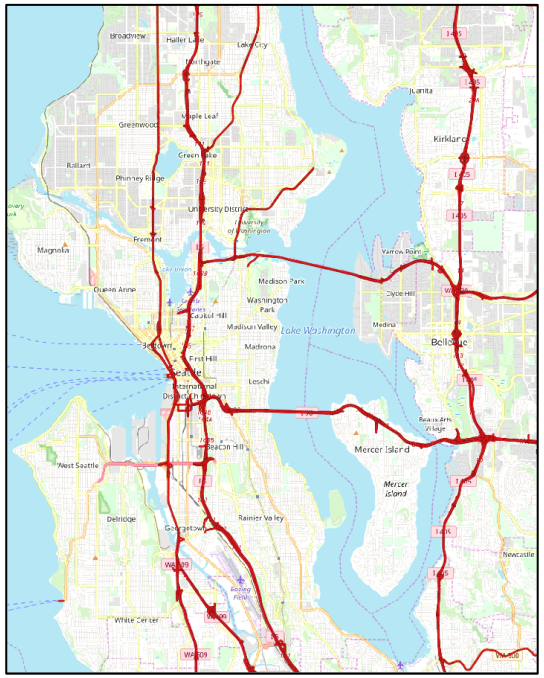
starting and ending points

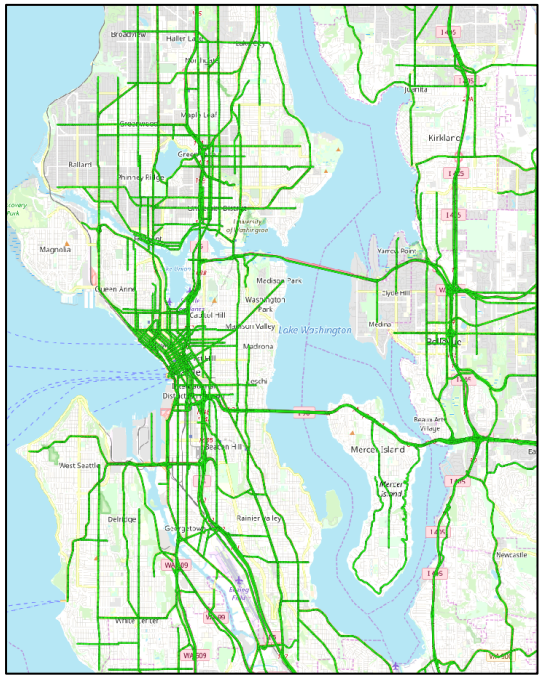
- Important to transportation data management
  - Calculating average travel time of the shortest path from Point A to Point B

Transportation Data Management

Example: SHRP2 Reliability Data Analysis and Tools

One of the task: Combining Loop detector data and HERE (like INRIX) data by conducting map conflation



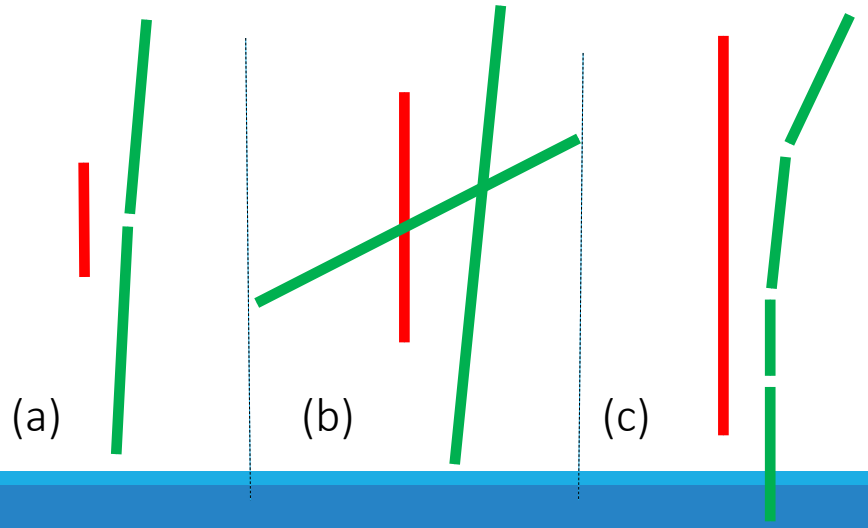


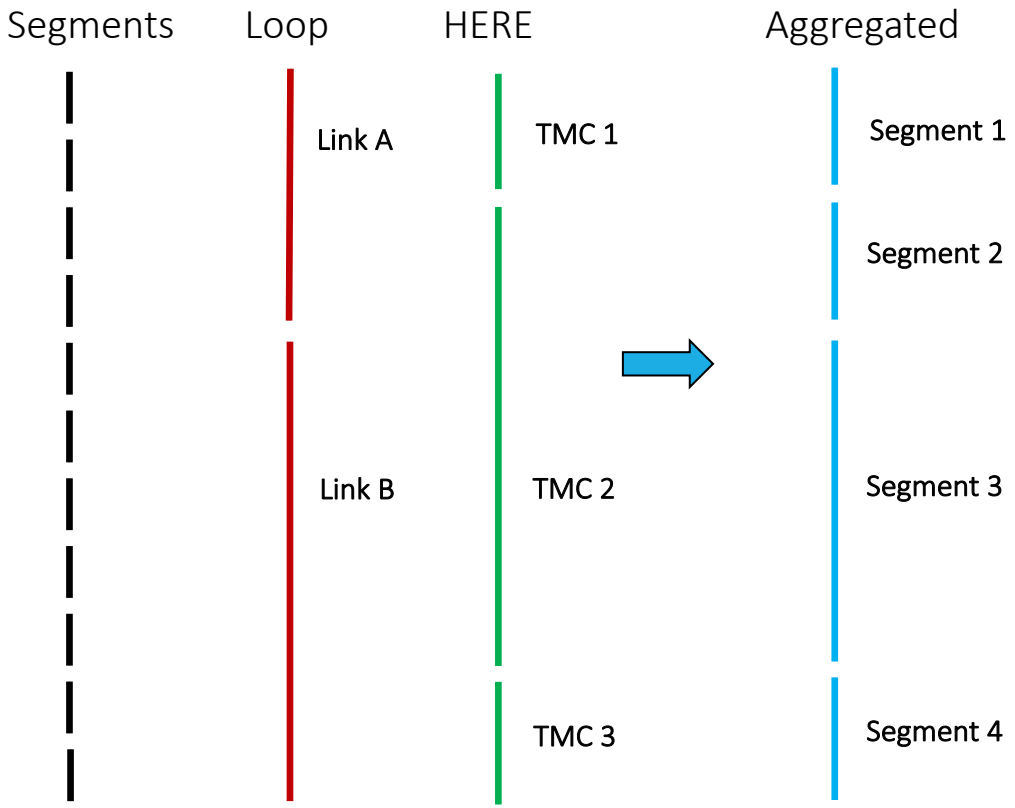
Loop Detector Road Layer
HERE Road Layer

Transportation Data Management

How to fulfill this task?

1. Store road links in PostgreSQL;
2. Split roads into small segments;
3. Use PostGIS to calculate distance and angles between segment in the two datasets.
4. Match the nearest pairs





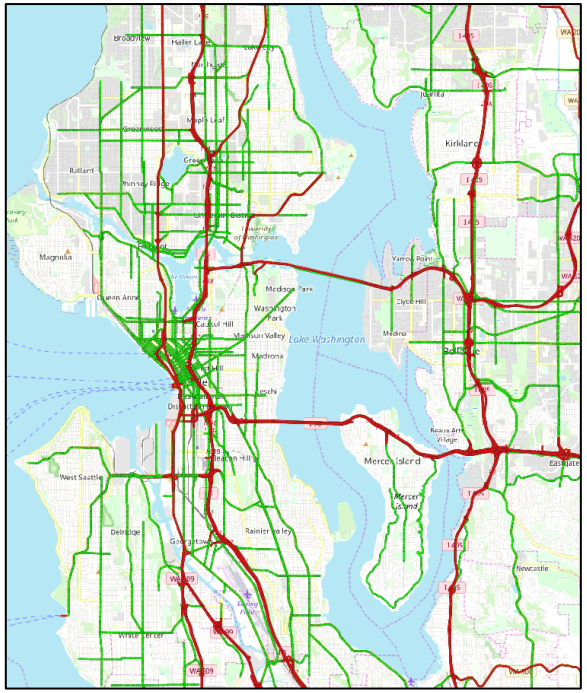
Transportation Data Management

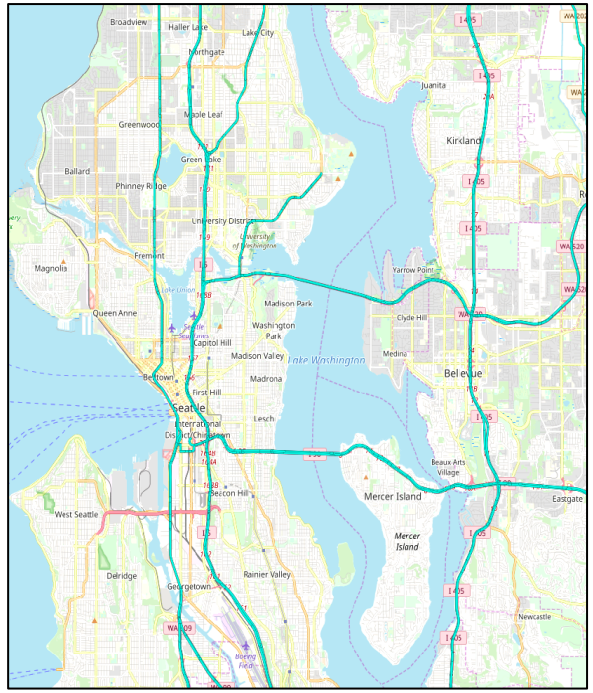
Spatial integration

- 10482 segments in loop detector data
- 28007 segments in HERE data
- ➢ 3692 matched segments

Spatial temporal integration

- using relational database, just like join tables based on spatial info and temporal info





Loop Detector Road Layer
HERE Road Layer
Combined/conflated Layer

QGIS

When you write PostgreSQL queries, you want to see how a curved line looks like. How to do that?

QGIS... https://www.qgis.org/en/site/

QGIS is a free and open-source cross-platform desktop geographic information system application that supports viewing, editing, and analysis of geospatial data.

QGIS can easily connect to PostgreSQL to view/edit your geospatial data.

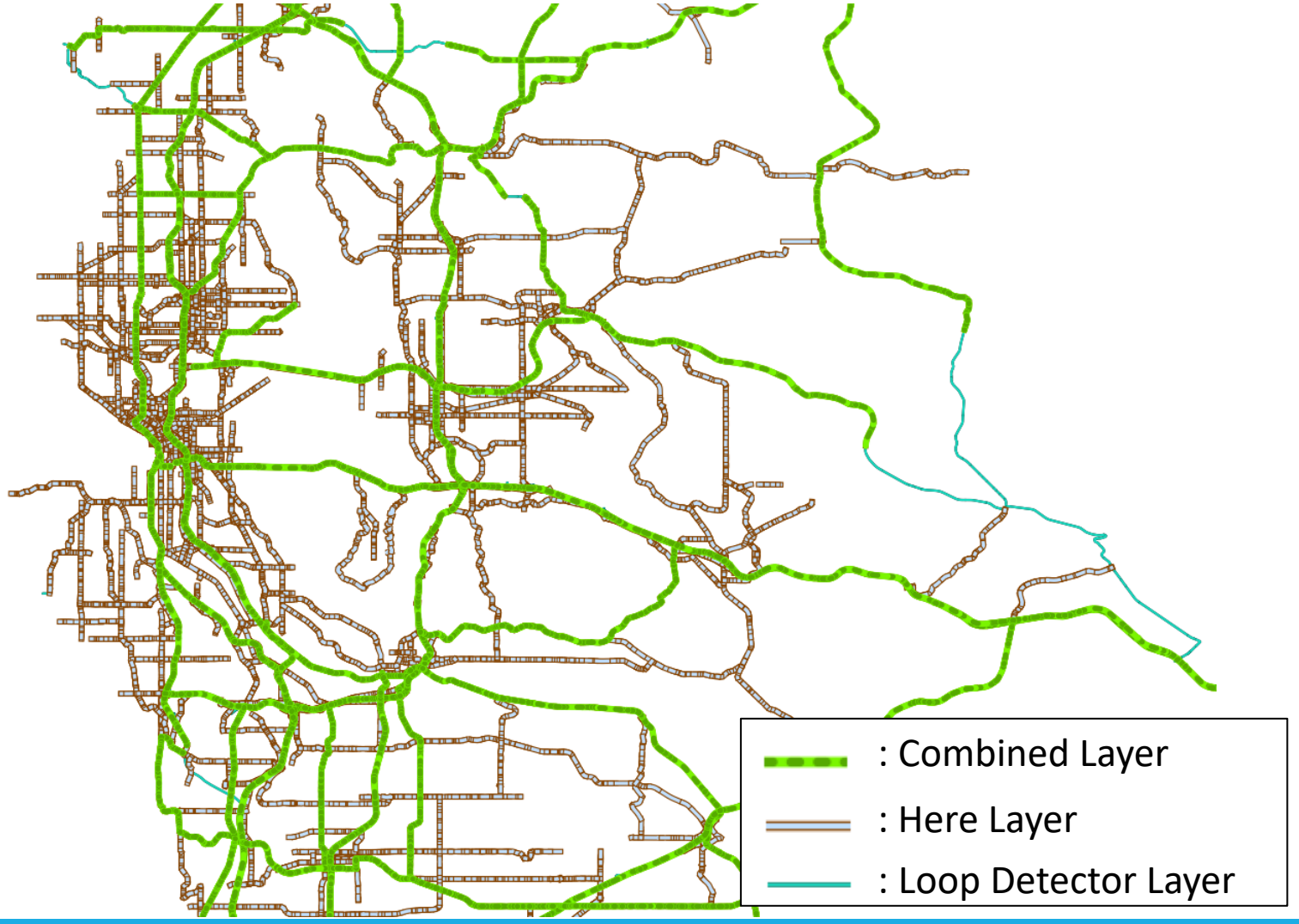
QGIS

Example:

- Visualization of the conflated roadway segment layers
- Editing geospatial data

Notes:

- QGIS is very helpful when you use PostgreSQL or PostGIS to process geospatial data.
- You can easily find more info/tutorials online.



NoSQL

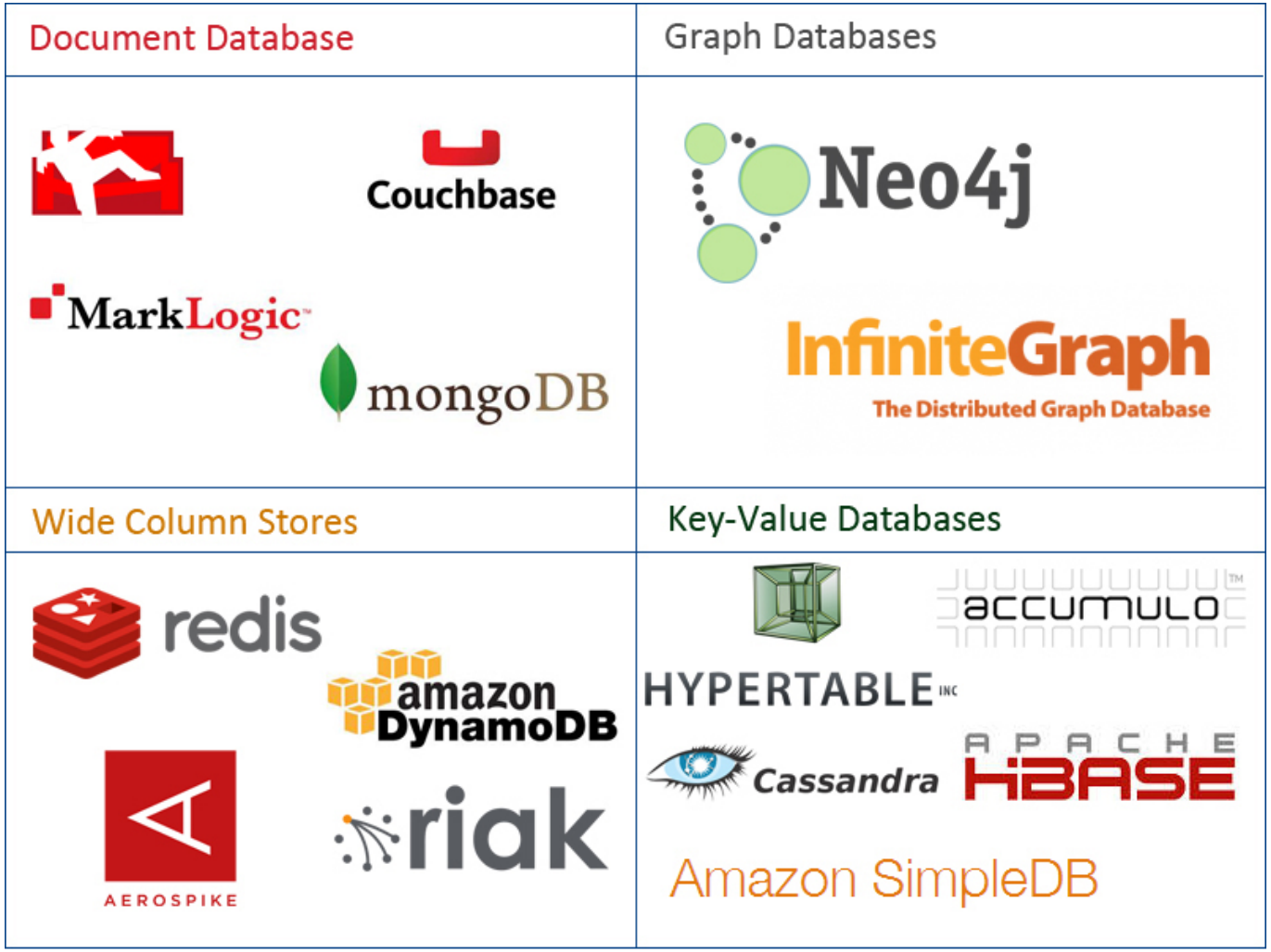
NoSQL: non SQL or non relational

- The data structures used by NoSQL databases are different from those used by default in relational databases, making some operations faster in NoSQL
  - Key-value
  - Wide column
  - Graph
  - Document

Sometimes, these data structures used by NoSQL databases are more flexible than relational database tables

So many NoSQL databases are on available now… How to choose?

NoSQL


Document Database
Couchbase
MarkLogic
mongoDB
Graph Databases
Neo4j
InfiniteGraph
The Distributed Graph Database
Wide Column Stores
redis
amazon DynamoDB
AEROSPIKE
riak
Key-Value Databases
accumulo
HYPERTABLE INC
Cassandra
APACHE HBASE
Amazon SimpleDB
@cloudtxt http://www.aryannava.com

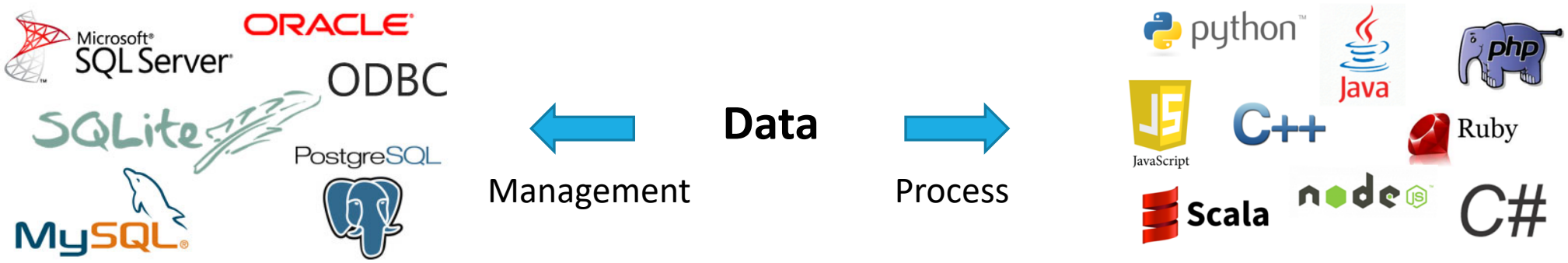
NoSQL

Transportation data management examples:

- Traffic real-time data → provided by API → key-value format → key-value database
- Traffic network → graph → graph database

Whether use NoSQL or not depends on your tasks.

In most cases, transportation data **process** and **management** are conducted at the same time or overlapped.



Summary

Transportation Data Management

- DBMS
- DB Design
- E/R Diagram
- SQL
- SQL Server and other database/tools

Not covered in this class:

- Cloud computing
- Distributed data process frameworks, such as Spark

Next step

- Transportation Data Analysis