TEKsystems
Own change

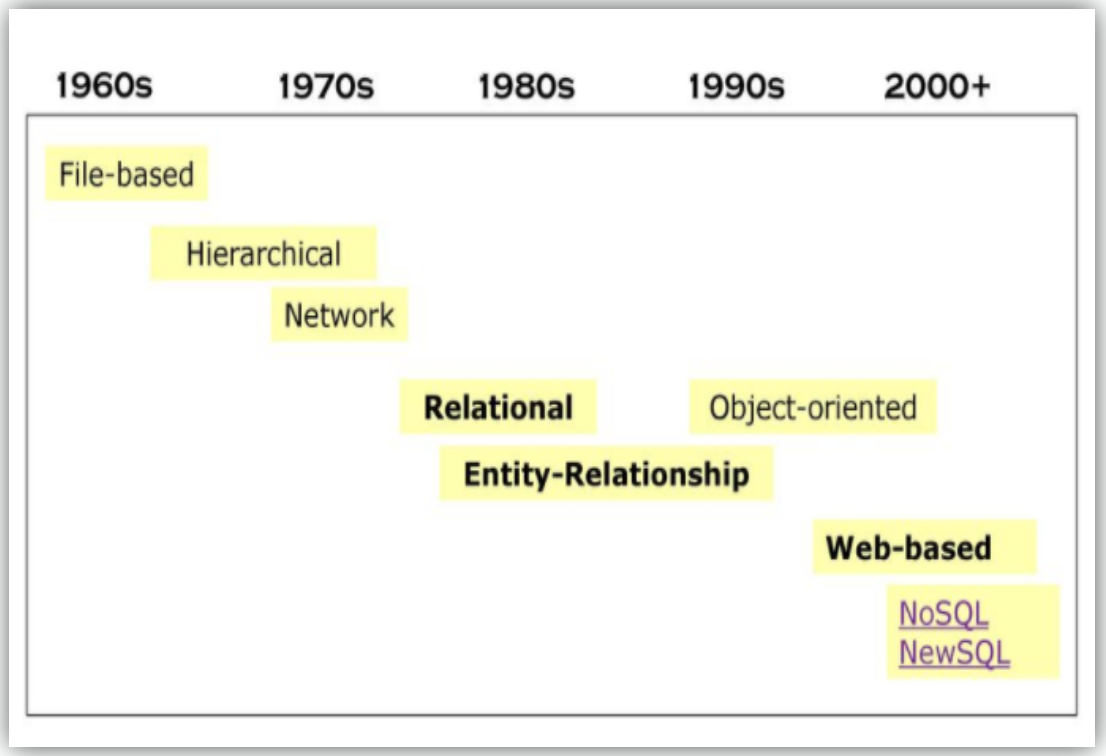
DATA ANALYTICS & INSIGHTS

Importance of SQL, PLSQL, DWH, Programming

Presented by: Vijay Guntumadugu & Vineesh Kaladharan

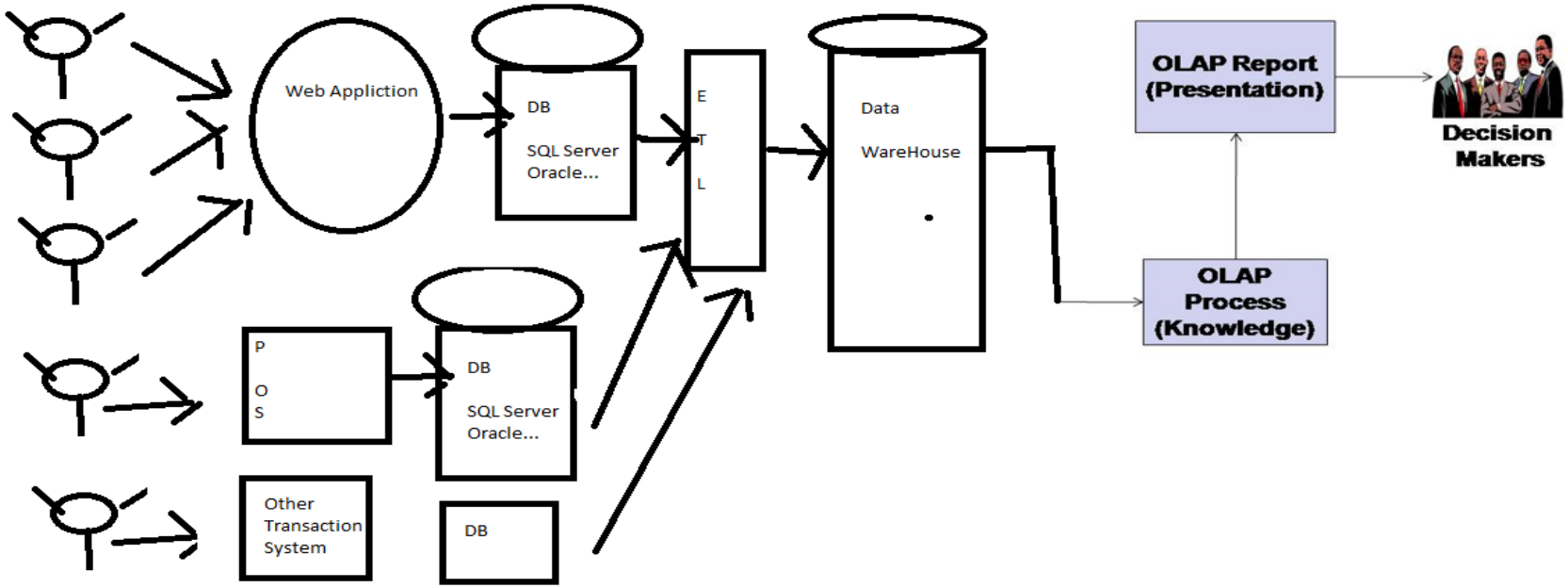
09-11-2020

SQL Evolution

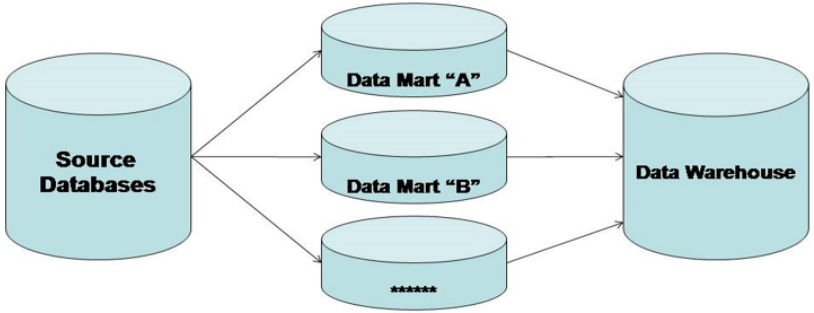


- Prerational databases did not have a set of commands to work with data.
- Every database either had its own proprietary language or used programs written in COBOL, C
- The databases were virtually inflexible and did not allow any internal structure changes without bringing them offline and rewriting tons of code.
- In the early 1970s, the growth of online applications (programs that require user's interaction) triggered the demand for something more flexible.

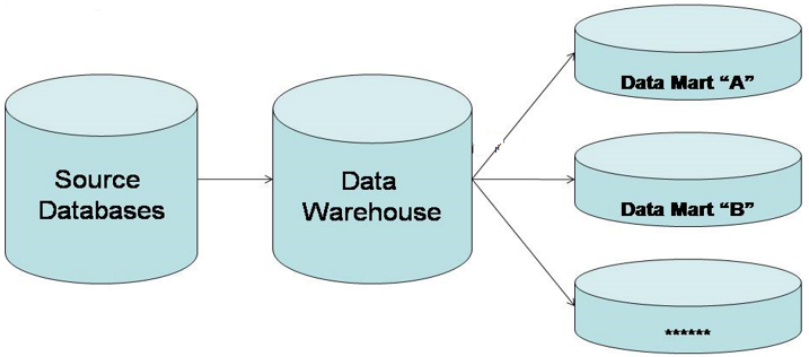
DWH & Importance of SQL



Bottom Up vs Top Down Approach



Top Down Approach



Bottom Up Approach

SQL: The Backbone of Data Analytics and Business Intelligence

The Foundation of BI

- SQL (Structured Query Language) serves as the base for a majority of the ETL tools and BI (Business Intelligence) Frameworks.
- Every Data Analyst starts interacting with the data by writing SQL Queries in order to get a good understanding of the relationships and the data attributes.
- While all the leading BI tools are designed to run against Oracle and MySQL databases, or even JSON, XML and Excel flat files, the core database system for many Windows OS business software applications is SQL.

SQL is a Simple and Declarative Language

- It can be termed as a natural language for analyzing datasets and underlying data sources because of the ease of understanding and simplicity of writing.
- Within the language of SQL these are common steps:
    1) projection statements (SELECT)
    2) filters and relational joins (WHERE)
    3) aggregation (GROUP BY).
- SQL is purely a Declarative Language i.e. we just need to declare or state the nature of the results that we would like to get from the query as a desired output.

SQL: Features

1. Data Definition Language
   - It contains commands used to define the data and structure of relational metadata.
   - Includes commands like CREATE, DROP, ALTER Tables primarily dealing with impacting the table structures.
2. Data Manipulation Language
   - It contains commands used to modify or manipulate the underlying data values within a relation.
   - Includes commands like INSERT, UPDATE, DELETE to insert new rows of data, delete existing rows and modify the values of any existing attributes.
3. Triggers
   - Triggers are side actions performed when certain conditions or requirements are met on the data.
   - Contains 3 major components such as Event (change in the data that activates the trigger), Condition (query executed because of the trigger) and Action (procedure executed when condition and event are true).
4. Transaction Control Language:
   - Contains important commands to control the DB transactions such as Commit, Rollback.
5. Client Server Execution and Remote Database Access

More about Select

- Select specific or all columns
- Distinct rows
- Filtering with where clause
- Wild Cards in SQL Server
- Joining multiple conditions using AND & OR operators
- Sorting rows using order by
- Selecting top n or top n percentage of rows

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
HAVING *condition*
ORDER BY *column_name(s);*

Types of Joins

- Inner Join
- Outer Joins
  - ✓ Left Outer Join
  - ✓ Right Outer join
  - ✓ Full Outer Join
- Cross Join

Joins in SQL Server are used to retrieve data from 2 or more related tables. In general tables are related to each other using foreign key constraints.

In SQL server, there are different types of JOINS

1. INNER JOIN
2. OUTER JOIN
3. CROSS JOIN

Outer Joins are again divided into

1. Left Join or Left Outer Join
2. Right Join or Right Outer Join
3. Full Join or Full Outer Join

Name	Gender	Salary	DepartmentName
Tom	Male	4000	IT
Pam	Female	3000	HR
John	Male	3500	IT
Sam	Male	4500	Payroll
Todd	Male	2800	Payroll
Ben	Male	7000	IT
Sara	Female	4800	HR
Valarie	Female	5500	IT

ID	Name	Gender	Salary	DepartmentId
1	Tom	Male	4000	1
2	Pam	Female	3000	3
3	John	Male	3500	1
4	Sam	Male	4500	2
5	Todd	Male	2800	2
6	Ben	Male	7000	1
7	Sara	Female	4800	3
8	Valarie	Female	5500	1
9	James	Male	6500	NULL
10	Russell	Male	8800	NULL

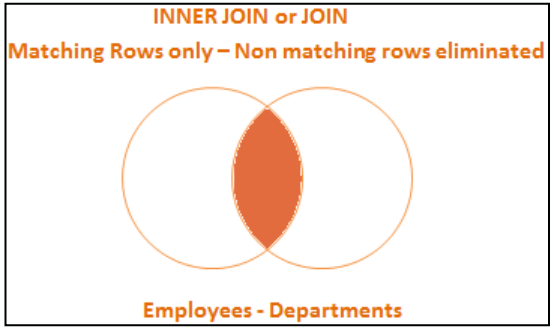
Id	DepartmentName	Location	DepartmentHead
1	IT	London	Rick
2	Payroll	Delhi	Ron
3	HR	New York	Christie
4	Other Department	Sydney	Cindrella

Inner Join or Join

ID	Name	Gender	Salary	DepartmentId
1	Tom	Male	4000	1
2	Pam	Female	3000	3
3	John	Male	3500	1
4	Sam	Male	4500	2
5	Todd	Male	2800	2
6	Ben	Male	7000	1
7	Sara	Female	4800	3
8	Valarie	Female	5500	1
9	James	Male	6500	NULL
10	Russell	Male	8800	NULL

Id	DepartmentName	Location	DepartmentHead
1	IT	London	Rick
2	Payroll	Delhi	Ron
3	HR	New York	Christie
4	Other Department	Sydney	Cindrella

Name	Gender	Salary	DepartmentName
Tom	Male	4000	IT
Pam	Female	3000	HR
John	Male	3500	IT
Sam	Male	4500	Payroll
Todd	Male	2800	Payroll
Ben	Male	7000	IT
Sara	Female	4800	HR
Valarie	Female	5500	IT



INNER JOIN returns only the matching rows between both the tables. Non matching rows are eliminated.

```
SELECT Name, Gender, Salary, DepartmentName
FROM tblEmployee
INNER JOIN tblDepartment
ON tblEmployee.DepartmentId = tblDepartment.Id
```

OR

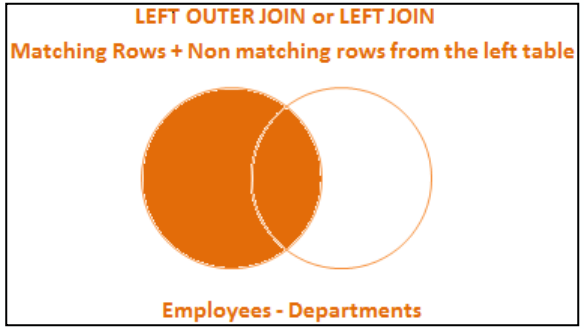
```
SELECT Name, Gender, Salary, DepartmentName
FROM tblEmployee
JOIN tblDepartment
ON tblEmployee.DepartmentId = tblDepartment.Id
```

Left Outer Join or Left Join

ID	Name	Gender	Salary	DepartmentId
1	Tom	Male	4000	1
2	Pam	Female	3000	3
3	John	Male	3500	1
4	Sam	Male	4500	2
5	Todd	Male	2800	2
6	Ben	Male	7000	1
7	Sara	Female	4800	3
8	Valarie	Female	5500	1
9	James	Male	6500	NULL
10	Russell	Male	8800	NULL

Id	DepartmentName	Location	DepartmentHead
1	IT	London	Rick
2	Payroll	Delhi	Ron
3	HR	New York	Christie
4	Other Department	Sydney	Cindrella

Name	Gender	Salary	DepartmentName
Name	Gender	Salary	DepartmentName
Tom	Male	4000	IT
Pam	Female	3000	HR
John	Male	3500	IT
Sam	Male	4500	Payroll
Todd	Male	2800	Payroll
Ben	Male	7000	IT
Sara	Female	4800	HR
Valarie	Female	5500	IT
James	Male	6500	NULL
Russell	Male	8800	NULL



LEFT JOIN returns all the matching rows + non matching rows from the left table.

```
SELECT Name, Gender, Salary, DepartmentName
FROM tblEmployee
LEFT OUTER JOIN tblDepartment
ON tblEmployee.DepartmentId = tblDepartment.Id
```

OR

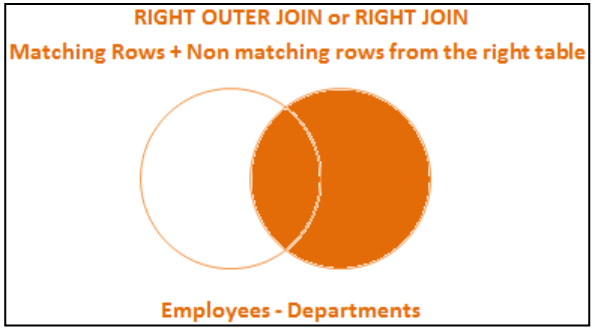
```
SELECT Name, Gender, Salary, DepartmentName
FROM tblEmployee
LEFT JOIN tblDepartment
ON tblEmployee.DepartmentId = tblDepartment.Id
```

Right Outer Join or Right Join

ID	Name	Gender	Salary	DepartmentId
1	Tom	Male	4000	1
2	Pam	Female	3000	3
3	John	Male	3500	1
4	Sam	Male	4500	2
5	Todd	Male	2800	2
6	Ben	Male	7000	1
7	Sara	Female	4800	3
8	Valarie	Female	5500	1
9	James	Male	6500	NULL
10	Russell	Male	8800	NULL

Id	DepartmentName	Location	DepartmentHead
1	IT	London	Rick
2	Payroll	Delhi	Ron
3	HR	New York	Christie
4	Other Department	Sydney	Cindrella

Name	Gender	Salary	DepartmentName
Tom	Male	4000	IT
John	Male	3500	IT
Ben	Male	7000	IT
Valarie	Female	5500	IT
Sam	Male	4500	Payroll
Todd	Male	2800	Payroll
Pam	Female	3000	HR
Sara	Female	4800	HR
NULL	NULL	NULL	Other Department



RIGHT JOIN returns all the matching rows + non matching rows from the right table

```
SELECT Name, Gender, Salary, DepartmentName
FROM tblEmployee
RIGHT OUTER JOIN tblDepartment
ON tblEmployee.DepartmentId = tblDepartment.Id
```

OR

```
SELECT Name, Gender, Salary, DepartmentName
FROM tblEmployee
RIGHT JOIN tblDepartment
ON tblEmployee.DepartmentId = tblDepartment.Id
```

Full Outer Join or Full Join

ID	Name	Gender	Salary	DepartmentId
1	Tom	Male	4000	1
2	Pam	Female	3000	3
3	John	Male	3500	1
4	Sam	Male	4500	2
5	Todd	Male	2800	2
6	Ben	Male	7000	1
7	Sara	Female	4800	3
8	Valarie	Female	5500	1
9	James	Male	6500	NULL
10	Russell	Male	8800	NULL

Id	DepartmentName	Location	DepartmentHead
1	IT	London	Rick
2	Payroll	Delhi	Ron
3	HR	New York	Christie
4	Other Department	Sydney	Cindrella

Name	Gender	Salary	DepartmentName
Tom	Male	4000	IT
Pam	Female	3000	HR
John	Male	3500	IT
Sam	Male	4500	Payroll
Todd	Male	2800	Payroll
Ben	Male	7000	IT
Sara	Female	4800	HR
Valarie	Female	5500	IT
James	Male	6500	NULL
Russell	Male	8800	NULL
NULL	NULL	NULL	Other Department

FULL OUTER JOIN or FULL JOIN

Matching Rows + Non matching rows from both the tables

Employees - Departments

FULL JOIN returns all rows from both the left and right tables, including the non matching rows.

```
SELECT Name, Gender, Salary, DepartmentName
FROM tblEmployee
FULL OUTER JOIN tblDepartment
ON tblEmployee.DepartmentId = tblDepartment.Id
```

OR

```
SELECT Name, Gender, Salary, DepartmentName
FROM tblEmployee
FULL JOIN tblDepartment
ON tblEmployee.DepartmentId = tblDepartment.Id
```

Cross Join

CROSS JOIN, produces the Cartesian product of the 2 tables involved in the join. For example, in the Employees table we have 10 rows and in the Departments table we have 4 rows. So, a cross join between these 2 tables produces 40 rows.

NOTE: Cross Join shouldn't have ON clause.

General Formula for Joins:

```
SELECT      COLUMN_LIST
FROM        LEFT_TABLE_NAME
JOIN_TYPE   RIGHT_TABLE_NAME
ON          JOIN_CONDITION
```

Cross Join:

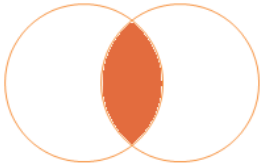
```
SELECT      Name, Gender, Salary, DepartmentName
FROM        tblEmployee
CROSS JOIN  tblDepartment
```

Summary

Join Type	Purpose
Cross Join	Returns Cartesian product of the tables involved in the join
Inner Join	Returns only the matching rows. Non matching rows are eliminated.
Left Join	Returns all the matching rows + non matching rows from the left table
Right Join	Returns all the matching rows + non matching rows from the right table
Full Join	Returns all rows from both tables, including the non-matching rows.

INNER JOIN or JOIN

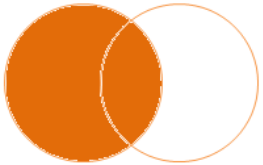
Matching Rows only – Non matching rows eliminated



Employees - Departments

LEFT OUTER JOIN or LEFT JOIN

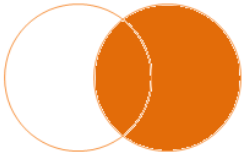
Matching Rows + Non matching rows from the left table



Employees - Departments

RIGHT OUTER JOIN or RIGHT JOIN

Matching Rows + Non matching rows from the right table



Employees - Departments

FULL OUTER JOIN or FULL JOIN

Matching Rows + Non matching rows from both the tables

Employees - Departments

Use of Programming Languages in DAI

The practice of data analytics and insights requires the use of programming languages to help data professionals extract insights and value from data.

- Data analysts use python to create machine learning algorithms, perform data mining, build data models, visualize data, create web services and classify data sets.

- R is a language used for statistical computations, data cleaning, data analysis and graphical representation of data.

Use of Programming Languages in DAI

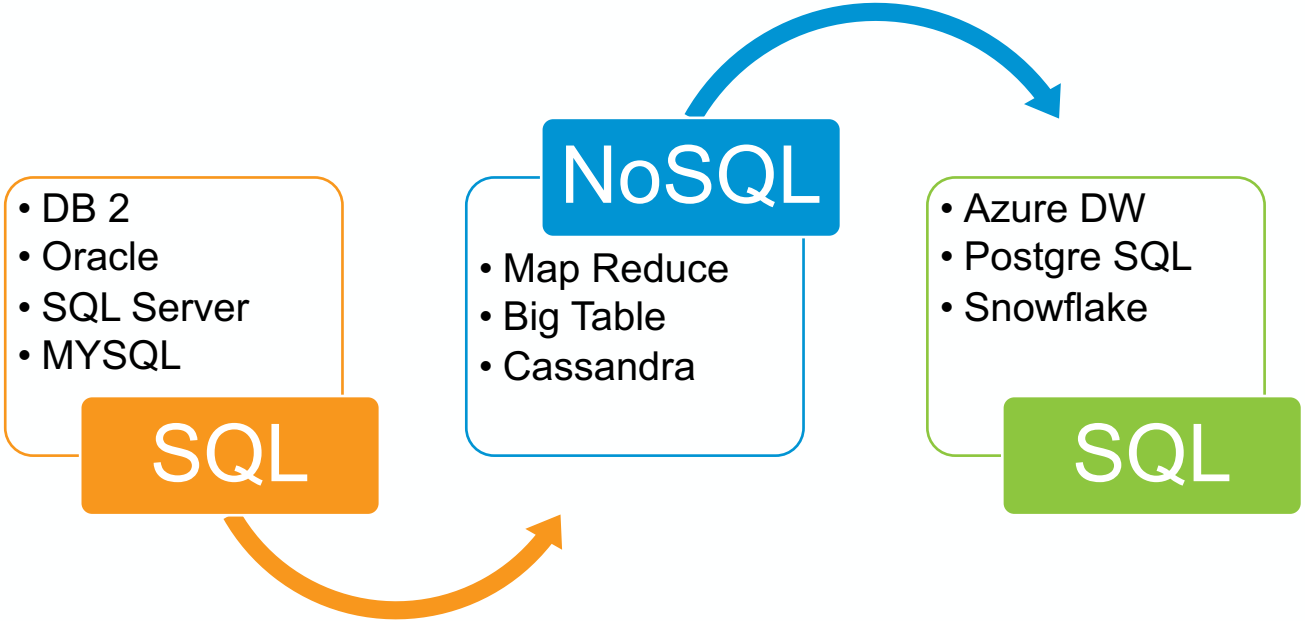
The practice of data analytics and insights requires the use of programming languages to help data professionals extract insights and value from data.

- Most of the popular frameworks and tools used for Big Data are typically written in Java and Scala. This includes Fink, Hadoop, Hive, Kafka and Spark.

- Scala is capable to work with the data which is stored in a Distributed fashion and perform parallel data processing.

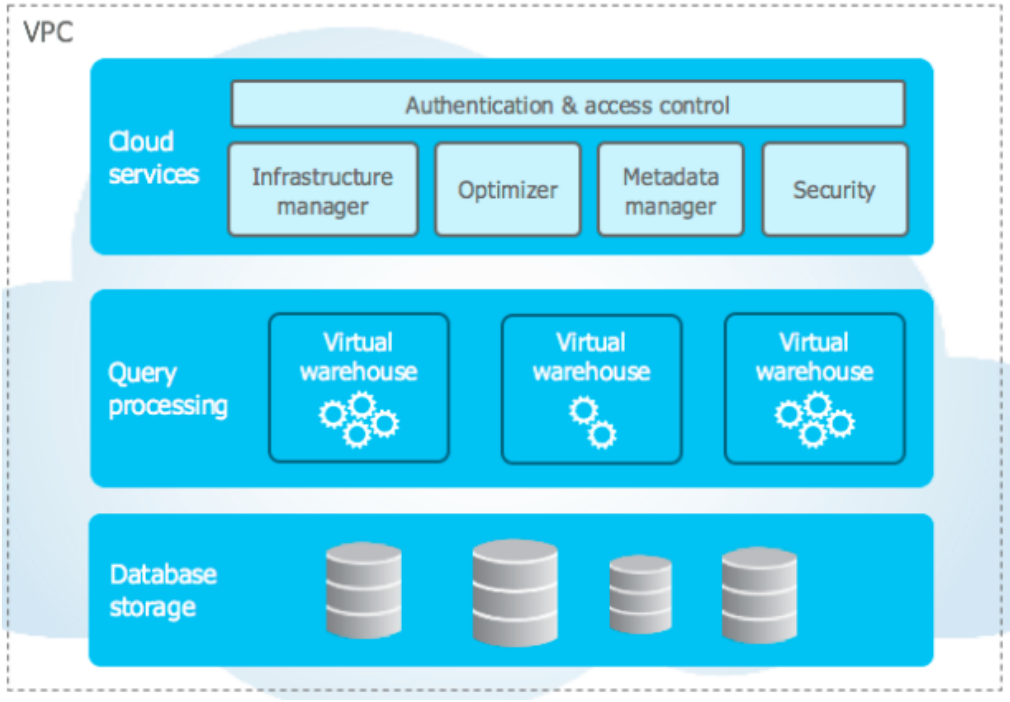
- .NET programmers use and re-use huge data sets to develop effective business logic and solutions with C#.

Journey of Data



Snowflake

- DWH on Cloud with less maintenance and it is not hosted in any application unlike big data which is housed on Hadoop
- Based on Native SQL
- Separates Storage and Compute and they can scale independently
- Time Travel
- Zero-copy cloning



Thank You