Master Executive di II Livello

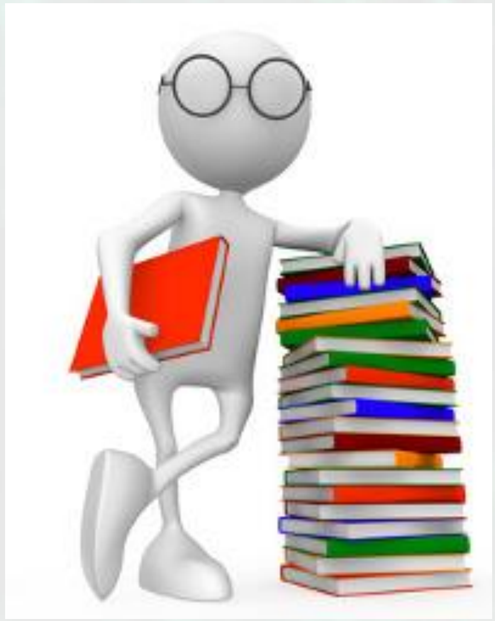
BIG DATA ANALYSIS AND BUSINESS INTELLIGENCE

Vamsi Krishna Varma Gunturi
Data science intern at ISTAT
vamsivarmagunturi@gmail.com

Databases and Hadoop

fondazione INUIT TORVERGATA

Todays topics



- Introduction to DBMS
- SQL
- Hive QL

What is DBMS ?

- The Database Management System (DBMS) is the software system responsible of managing the database. Data in the database are accessible only through such system.
- A database is a collection of inter-related data organized in particular ways, and managed by a DBMS.
- A database management system (DBMS) is a set of programs that allows one to carry out at least the following tasks:
  - Create a (persistent) database.
  - Insert, delete, modify (update) data in a database.
  - Query a database **efficiently** (extract information).
  - Ensuring **correctness** and **availability** in data management

Applications of DBMS

- Traditional applications:
  - Institutional records
    - Government, Corporate, Academic, …
    - Payroll, Personnel Records, …
  - Airline Reservation Systems
  - Banking Systems

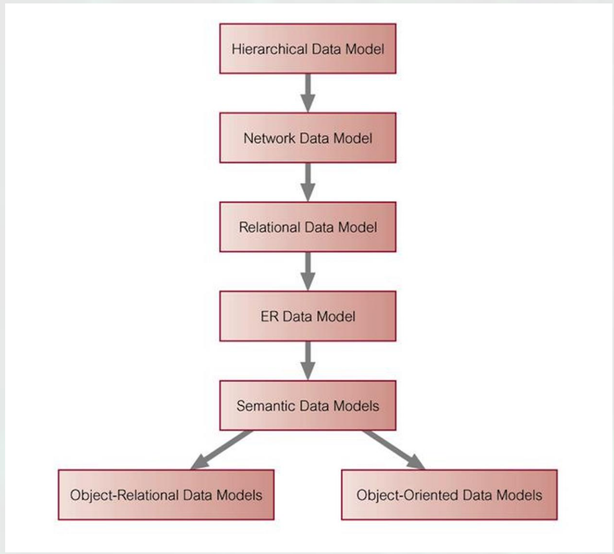
- Numerous new applications:
  - Scientific Databases
  - Electronic Health Records
  - Information Integration from Heterogeneous Sources
  - Databases are behind most of the things on the web:
    Google searches, Amazon purchases, eBay auctions, …

Data Languages

A Data Language has two parts:

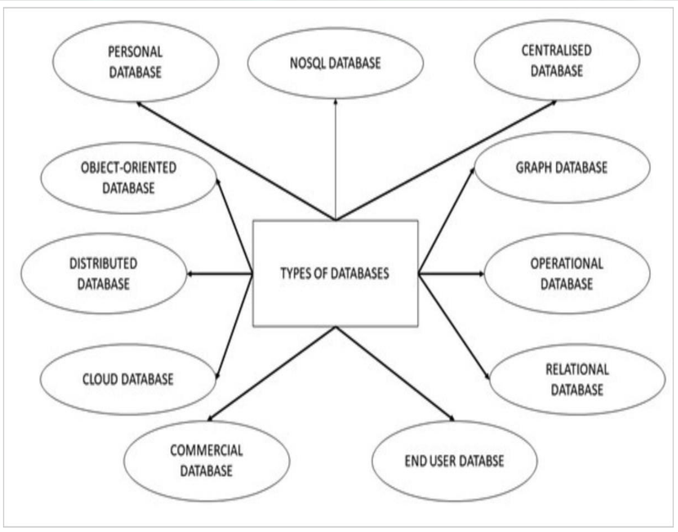
- A Data Definition Language (DDL) has a syntax for describing **database templates** in terms of the underlying data model.
- A Data Manipulation Language (DML) supports the following operations on data:
  - Insertion
  - Deletion
  - Update
  - Retrieval and extraction of data (query the data).

History of Data Models



- Earlier Data Models (before 1970)
  - Hierarchical Data Model
    - Based on the mathematical notion of a tree.
  - Network Data Model
    - Based on the mathematical notion of a graph.
- Relational Data Model – 1970
  - Based on the mathematical notion of a relation.
- Entity-Relationship Model – 1976
  - Conceptual model; used mainly as a design tool.
- Semi-structured Data Model and XML – late 1990s
  - Based on SGML and the mathematical notion of a tree (the Hierarchical Model strikes back!).
- Data Model of Graph-databases – 2000s

Types of databases



- Centralised database.
- Distributed database.
- Commercial database.
- NoSQL database.
- Operational database.
- Relational database.
- Cloud database.
- Object-oriented database.
- Graph database.

SQL topics

- Basic SQL statements
- Restricting and sorting data
- Displaying data with multiple tables
- Aggregating data using group functions
- Manipulating data
- Creating and managing tables
- Creating views and indexes

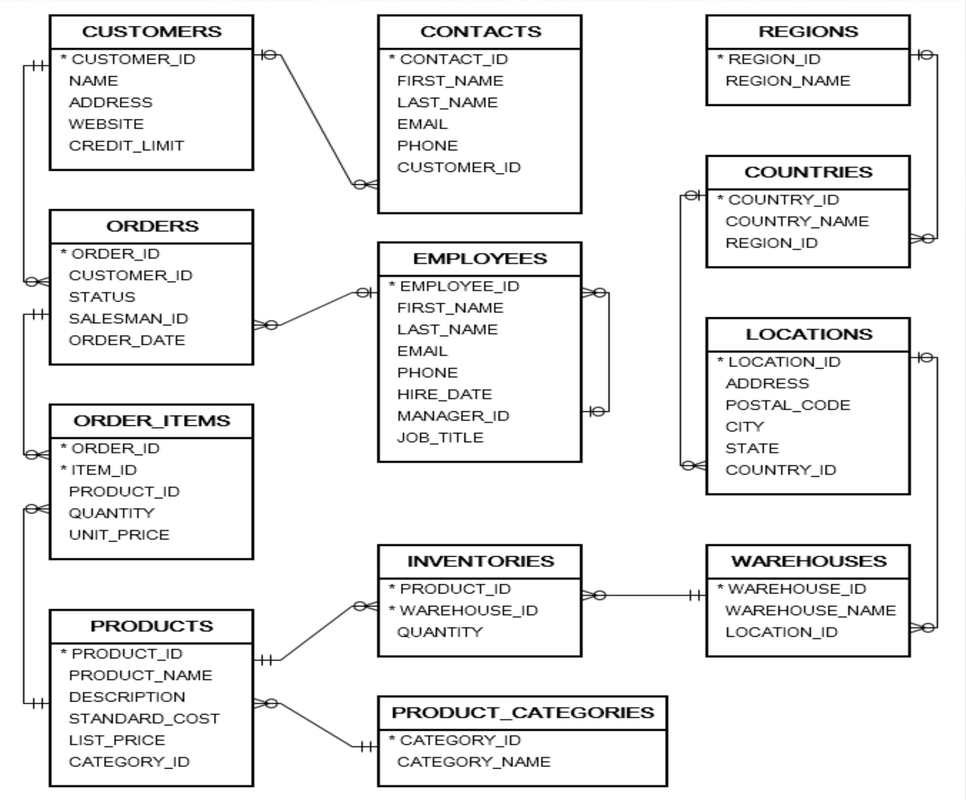
What is SQL ?

- SQL is the standard language for relational DBMSs
- RDBMS: A database management system that manages data as a collection of tables in which all relationships are represented by common values in related tables
- Specify syntax/semantics for data definition and manipulation

SQL statements

- SQL statements are not case sensitive.
- SQL statements can be on one or more lines.
- Keywords cannot be abbreviated or split across lines.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.

SQL Schema


CUSTOMERS
* CUSTOMER_ID
NAME
ADDRESS
WEBSITE
CREDIT_LIMIT
CONTACTS
* CONTACT_ID
FIRST_NAME
LAST_NAME
EMAIL
PHONE
CUSTOMER_ID
REGIONS
* REGION_ID
REGION_NAME
ORDERS
* ORDER_ID
CUSTOMER_ID
STATUS
SALESMAN_ID
ORDER_DATE
EMPLOYEES
* EMPLOYEE_ID
FIRST_NAME
LAST_NAME
EMAIL
PHONE
HIRE_DATE
MANAGER_ID
JOB_TITLE
COUNTRIES
* COUNTRY_ID
COUNTRY_NAME
REGION_ID
LOCATIONS
* LOCATION_ID
ADDRESS
POSTAL_CODE
CITY
STATE
COUNTRY_ID
ORDER_ITEMS
* ORDER_ID
* ITEM_ID
PRODUCT_ID
QUANTITY
UNIT_PRICE
INVENTORIES
* PRODUCT_ID
* WAREHOUSE_ID
QUANTITY
WAREHOUSES
* WAREHOUSE_ID
WAREHOUSE_NAME
LOCATION_ID
PRODUCTS
* PRODUCT_ID
PRODUCT_NAME
DESCRIPTION
STANDARD_COST
LIST_PRICE
CATEGORY_ID
PRODUCT_CATEGORIES
* CATEGORY_ID
CATEGORY_NAME

Examples

- Select all the rows from a table –

```
SELECT *
FROM EMPLOYEES;
```

- Select a specific employee -

```
SELECT *
FROM EMPLOYEES
WHERE first_name = ”Francesco”;
```

- Ordering employees alphabetically in descending order based on their last name –

```
SELECT *
FROM EMPLOYEES
ORDER BY last_name DESC;
```

Restricting and sorting data

Basic structure of the query -

```
SELECT *|{[DISTINCT] column | expression [alias],...}
FROM table
[WHERE condition(s)]
[ORDER BY {column, expr, alias} [ASC|DESC]];
```

Restricting and sorting data (2)

- Using WHERE clause –

```
SELECT employee_id, last_name, job_id, department_id
FROM EMPLOYEES
WHERE department_id = 90 ;
```

- Using comparison conditions –

```
SELECT last_name, salary
FROM EMPLOYEES
WHERE salary <= 3000;
```

Restricting and sorting data (3)

- Use the BETWEEN, IN, LIKE, and NULL conditions –

```
SELECT last_name, salary
FROM EMPLOYEES
WHERE salary BETWEEN 2500 AND 3500;

SELECT employee_id, last_name, salary, manager_id
FROM EMPLOYEES
WHERE manager_id IN (100, 101, 201);

SELECT first_name
FROM EMPLOYEES
WHERE first_name LIKE 'S%';
```

Restricting and sorting data (3)

- Apply the logical AND, OR, and NOT operators –

```
SELECT employee_id, last_name, job_id, salary
FROM EMPLOYEES
WHERE salary >=10000 AND job_id LIKE '%MAN%';

SELECT employee_id, last_name, job_id, salary
FROM EMPLOYEES
WHERE salary >= 10000 OR job_id LIKE '%MAN%';

SELECT last_name, job_id
FROM EMPLOYEES
WHERE job_id NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP');
```

Restricting and sorting data (4)

- Use the ORDER BY clause to sort rows of output –

```
SELECT last_name, job_id, department_id, hire_date
FROM EMPLOYEES
ORDER BY hire_date ;
```

- Sorting by Column alias –

```
SELECT employee_id, last_name, salary*12 annsal
FROM EMPLOYEES
ORDER BY annsal;
```

```
SELECT last_name, department_id, salary
FROM EMPLOYEES
ORDER BY department_id, salary DESC;
```

Displaying data with multiple tables

- Use a join to query data from more than one table -
  SELECT table1.column, table2.column
  FROM table1, table2
  WHERE table1.column1 = table2.column2;

- Types of joins:
  - Equi join
  - Non-equi join
  - Outer join
  - Self join

Displaying data with multiple tables (2)

- **EQUI Join** –

```
SELECT e.employee_id, e.last_name,
       e.department_id, d.department_id, d.location_id
FROM EMPLOYEES e, DEPARTMENTS d
WHERE e.department_id = d.department_id;
```

- **Non-EQUI Join** – Salary in the EMPLOYEES table must be between lowest salary and highest salary in the JOB_GRADES table.

```
SELECT e.last_name, e.salary, j.grade_level
FROM EMPLOYEES e, JOB_GRADES j
WHERE e.salary BETWEEN
j.lowest_sal AND j.highest_sal;
```

Displaying data with multiple tables (2)

- **LEFT OUTER JOIN** - returns all the rows from the left table, even if there are no matches in the right table

```
SELECT e.employee_id, e.last_name,
          e.department_id, d.department_id
FROM EMPLOYEES e  LEFT OUTER JOIN DEPARTMENTS d
ON (e.department_id = d.department_id);
```

- **RIGHT OUTER JOIN –** returns all the rows from the right table, even if there are no matches in the left table.

```
SELECT e.employee_id, e.last_name,
          e.department_id, d.department_id
FROM EMPLOYEES e RIGHT OUTER JOIN DEPARTMENTS d
ON (e.department_id = d.department_id);
```

Aggregating data using group functions

- Group functions operate on sets of groups to give one result per group
- Types of group functions – AVG, COUNT, MAX, MIN, STDDEV, SUM, VARIANCE
- Syntax:

```
SELECT [column,] group_function(column), ...
FROM table
[WHERE condition]
[GROUP BY column]
[ORDER BY column];
```

Aggregating data using group functions (2)

- Using the AVG and SUM Functions –

```
SELECT AVG(salary), MAX(salary),
            MIN(salary), SUM(salary)
FROM EMPLOYEES
WHERE job_id LIKE '%REP%';
```

- COUNT(*) returns the number of rows in a table –

```
SELECT COUNT(*)
FROM EMPLOYEES
WHERE department_id = 50;
```

Aggregating data using group functions (3)

- Using GROUP BY clause –

```
SELECT AVG(salary)
FROM employees
GROUP BY department_id ;
```

- Using HAVING clause –

```
SELECT department_id, MAX(salary)
FROM employees
GROUP BY department_id
HAVING MAX(salary)>10000 ;
```

Data Manipulation Language

A DML statement is executed when you:

- Add new rows to a table
- Modify existing rows in a table
- Remove existing rows from a table

Data Manipulation Language (2)

- Inserting rows in to the table -

```
INSERT INTO
DEPARTMENTS (department_id, department_name)
VALUES (30, 'Purchasing’);
```

- Updating rows in a table –

```
UPDATE EMPLOYEES
SET department_id = 55
WHERE department_id = 110;
```

- Deleting rows from a table –

```
DELETE FROM departments
WHERE department_name = 'Finance';
```

Database Objects

Object	Description
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Numeric value generator
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

Column datatypes

Data Type	Description
VARCHAR2(*size*)	Variable-length character data
CHAR(*size*)	Fixed-length character data
NUMBER(*p*,*s*)	Variable-length numeric data
DATE	Date and time values
LONG	Variable-length character data up to 2 gigabytes
CLOB	Character data up to 4 gigabytes
RAW and LONG RAW	Raw binary data
BLOB	Binary data up to 4 gigabytes
BFILE	Binary data stored in an external file; up to 4 gigabytes
ROWID	A 64 base number system representing the unique address of a row in its table.

Creating and managing tables

- Creating a new table –

```
CREATE TABLE dept
 (
   deptno NUMBER(2),
   dname VARCHAR2(14),
   loc VARCHAR2(13)
 );
```

Master Executive di II Livello

BIG DATA ANALYSIS AND BUSINESS INTELLIGENCE

Vamsi Krishna Varma Gunturi
Data science intern at ISTAT
vamsivarmagunturi@gmail.com

Grazie

fondazione INUIT TORVERGATA