**Oracle Database Management System (DBMS)**

**Group 1 :   
Harish Rao Yadagiri  
Anoop Garige  
Lakshmi Priya Chintada  
Sai Ramakanth Kodi**

1. **A detailed description of the DBMS**

**Overview :**

Oracle Database is a highly regarded Relational Database Management System (RDBMS) developed by Oracle Corporation. It is known for its robustness, scalability, and advanced features, making it one of the most popular database systems in the world.

**Key Features and Components :**

1. **Data Model**:
   * Oracle Database follows a relational data model, organizing data into tables with rows and columns.
   * It supports SQL (Structured Query Language) for data manipulation and retrieval.
2. **Architecture**:
   * Oracle's multi-tiered architecture:
     + **Client Tier**: Applications and users interact with the database.
     + **Application Tier**: Middleware and application servers manage application logic.
     + **Database Tier**: The actual Oracle Database resides in this tier, comprising:
       - **Instance**: Background processes and the System Global Area (SGA) for managing data in memory.
       - **Data Files**: Physical files on disk storing the actual data.
       - **Control Files**: Metadata and control information about the database.
       - **Redo Logs**: Record changes for recovery and consistency.
3. **Scalability**:
   * Oracle Database is highly scalable and supports clustering, partitioning, and replication for scalability and fault tolerance.
4. **Security**:
   * Robust security features include user authentication, authorization, and encryption.
   * Fine-grained access control and auditing ensure compliance with security regulations.
5. **Data Integrity**:
   * Data integrity is enforced through constraints, triggers, and referential integrity to maintain data accuracy and consistency.
6. **Data Types**:
   * Oracle supports a wide range of data types, including numeric, character, date and time, LOB, and user-defined types.
7. **Concurrency Control**:
   * Multi-Version Concurrency Control (MVCC) manages concurrent access to data, allowing multiple users to work simultaneously.
8. **Transaction Management**:
   * Oracle supports ACID properties for transaction reliability and consistency.
   * Transactions can be committed, rolled back, or savepoints set for fine-grained control.
9. **High Availability**:
   * Features like Data Guard, Real Application Clusters (RAC), and automatic failover ensure high availability and data redundancy.
10. **Backup and Recovery**:
    * Mechanisms for backup and recovery include RMAN (Recovery Manager), Flashback technology, and point-in-time recovery.
11. **Performance Tuning**:
    * Tools like AWR and ASH assist in monitoring and optimizing database performance.
12. **Partitioning and Compression**:
    * Data partitioning and compression options are available to efficiently manage large datasets.
13. **Data Warehousing and Business Intelligence**:
    * Oracle Database is commonly used for data warehousing and business intelligence due to its support for complex queries and reporting.
14. **PL/SQL**:
    * Oracle's proprietary language, PL/SQL, is used for developing stored procedures, functions, and triggers within the database.
15. **Cloud Integration**:
    * Oracle offers cloud-based versions of its database and tools for easy migration to the cloud.
16. **Extensibility**:
    * Custom data types, functions, and procedures can be created to extend the database's capabilities.

**Advantages of Oracle over other DBMS:**

**Oracle Database**, as a widely utilized relational database management system (RDBMS), boasts several strengths when compared to alternative database systems such as MySQL, PostgreSQL, SQLite, IBM Db2 on Cloud, Microsoft Azure SQL, MariaDB, Amazon Redshift, Redis, Google BigQuery, Google AlloyDB for PostgreSQL, Amazon Aurora, and SQLite. The following are key advantages of Oracle Database:

1. **Scalability**: Oracle Database stands out for its capability to manage substantial and enterprise-level workloads effectively. It offers features for partitioning, clustering, and data sharding, making it a fitting choice for high-demand applications.

2. **High Availability**: Oracle provides a range of features and technologies to achieve high availability, including Oracle Real Application Clusters (RAC), Data Guard, and automated failover capabilities, ensuring minimal downtime.

3. **Security**: Oracle Database implements a robust security model with elements like encryption, auditing, and fine-grained access control. It adheres to various security standards and regulations.

4. **Advanced Analytics**: Oracle Database supports advanced analytics and data processing through features such as in-database analytics, machine learning, and spatial data capabilities.

5. **Integration**: Oracle Database seamlessly integrates with other Oracle products and technologies, making it well-suited for organizations with a comprehensive Oracle ecosystem.

6. **Enterprise Support**: Oracle offers extensive enterprise-level support and services, making it a dependable choice for organizations requiring top-tier support and assistance.

7. **Data Warehousing**: Oracle Database offers features for constructing and managing data warehouses, positioning it as a strong contender for business intelligence and analytics applications.

8. **Advanced Query Optimization**: Oracle Database possesses a sophisticated query optimizer that efficiently handles complex queries, crucial for performance-critical applications.

9. **Compatibility**: Oracle Database supports standard SQL and provides compatibility features for migrating from other database systems.

10. **Backup and Recovery**: Oracle Database presents robust backup and recovery solutions, ensuring data protection and disaster recovery capabilities.

**OLTP in Oracle:**

Oracle is widely employed in Online Transaction Processing (OLTP) systems, which are typified by a multitude of short, online transactions encompassing INSERT, UPDATE, and DELETE operations. The primary focus in OLTP systems revolves around swift query processing, maintaining data integrity in multi-access environments, and evaluating effectiveness based on transaction throughput.

Oracle achieves the critical need for fast transaction processing in OLTP databases by harnessing in-memory data storage, streamlined network protocols, and effective transaction management. Oracle's architectural design segregates transaction processing from query processing, allowing concurrent processing without locking users out of the system.

Oracle adheres to the ACID (Atomicity, Consistency, Isolation, Durability) transaction model, ensuring reliable processing of all transactions. Concurrent transaction processing doesn't compromise data integrity. Oracle employs Multi-Version Concurrency Control (MVCC) to maintain consistent snapshots for read operations, even when data alterations are in progress, ensuring the stability of OLTP operations.

In summary, Oracle's combination of performance, scalability, reliability, security, and comprehensive support for transaction processing positions it as a top choice for enterprises in need of an efficient and dependable OLTP database system.

# A detailed description of the KDD Nuggets referenced data

We chose 2 datasets from KKD nuggets for the project

#### In the endeavor to construct a robust e-commerce transactional database, our focal point revolves around two primary datasets: 'Customers.csv' and 'TransactionalData.csv.' These datasets hold paramount importance in shaping the foundation of a comprehensive and efficient e-commerce database system. Our key objective is to establish a strong and interdependent relationship between these datasets, a critical step in ensuring the seamless functionality and effectiveness of our e-commerce database.

#### Dataset 1: Customer Dataset

#### Source Link: <https://www.kaggle.com/datasets/datascientistanna/customers-dataset>

Last modified date: 02/15/2023

Data Type: Behavioral and demographical

#### Data Description:

Shop Customer Data is a detailed analysis of an imaginative shop's ideal customers. It helps a business to better understand its customers. The owner of a shop gets information about Customers through membership cards.

|  |  |  |
| --- | --- | --- |
| Feature | Type of  Variable | Values |
| Customer | Nominal | 6-digit integer |
| Gender | Nominal | Strings(names) |
| Age | Numeric | Integer Values |
| Annual Income | Numeric | Integer values |
| Spending Score | Numeric | Integer values |
| Profession | Nominal | Strings(names) |
| Work Experience | Numeric | Integer Values |
| Family Size | Numeric | Integer Values |

#### Dataset Characteristics:

Multivariate: The dataset contains multiple variables.

Sequential: The data is organized sequentially, likely based on customer information.

Subject Area: Business

Dataset Size:

Number of Instances: 2001 Number of Features: 8

Additional Information: The dataset does not contain missing values. Variables:

1. Customer ID: A unique identifier assigned to each customer, serving as a distinct reference for individual records.

2. Gender: Indicates the customer's gender, providing information about whether they identify as male, female, or potentially another gender category.

3. Age: Represents the age of the customer, offering insight into the demographic distribution of age groups within the dataset.

4. Annual Income: Reflects the yearly earnings of the customer, providing a quantitative measure of their financial standing.

5. Spending Score: A numerical score assigned to each customer, ranging from 1 to 100, indicating their propensity to spend. Higher scores suggest a greater likelihood of spending.

6. Profession: Specifies the occupation or professional category of the customer, offering insights into their employment status and industry.

7. Work Experience: Provides information about the number of years of work experience the customer has accumulated, contributing to the understanding of their professional background.

8. Family Size: Indicates the size of the customer's family, offering insights into their household composition and potentially influencing spending behaviors.

#### Dataset 2:

Source Link: [https://www.kaggle.com/datasets/vipin20/transaction-data](https://www.kaggle.com/datasets/vipin20/transaction-data%20)

Data Type: Transactional

#### Data Description:

#### This dataset provides comprehensive insights into item purchase transactions with its eight key columns. The "UserId" serves as a unique identifier for each user, while "TransactionId" uniquely identifies each transaction. "TransactionTime" records the timestamp of each transaction, adding a temporal dimension to the data. "ItemCode" corresponds to the unique code assigned to purchased items, complemented by "ItemDescription" which offers a concise description of the items. The "NumberOfItemPurchased" column quantifies the total items bought in each transaction, and "CostPerItem" details the cost associated with each item. Finally, the "Country" column specifies the geographical location where the item was purchased, providing valuable context to the dataset.

#### Dataset Characteristics:

Multivariate: The dataset contains multiple variables.

Sequential: The data is organized sequentially, likely based on transaction timestamps.

Time-Series: The dataset involves time-series data, as it captures transactions over a period.

Subject Area: Business

**Dataset Size:**

Number of Instances: 22220

Number of Features: 6

Additional Information: The dataset does not contain missing values.

Variables:

**InvoiceNo:** This categorical variable serves as an ID for each transaction. It is a 6-digit integral number uniquely assigned to each transaction. Transactions starting with the letter 'c' indicate cancellations.

**StockCode:** Another categorical variable representing a product (item) code. It is a 5-digit integral number uniquely assigned to each distinct product.

**Description:** This is a categorical variable containing the product name.

**Quantity:** An integer feature representing the quantities of each product (item) per transaction. **InvoiceDate:** A date feature indicating the day and time when each transaction was generated. **UnitPrice:** A continuous feature representing the product's price per unit in sterling.

**CustomerID:** A categorical variable, a 5-digit integral number uniquely assigned to each customer.

**Country:** Another categorical variable representing the name of the country where each customer resides.

**Variables:**

|  |  |  |
| --- | --- | --- |
| Feature | Type of  Variable | Values |
| CustomerID | Numeric | integer |
| TransactionId | Numeric | integer |
| TransactionTime | Nominal | Strings  (gender) |
| ItemCode | Numeric | float |
| ItemDescription | Numeric | int |
| Number of Items Purchased | Numeric int | |
| CostPerItem | Numeric int | |
| Country | Nominal Strings | |

#### CustomerID: Uniquely identifies each customer, allowing for individual customer tracking within the transactions.

#### TransactionID: Represents a unique identifier for each transaction, facilitating the differentiation of various purchases.

#### TransactionTime: Captures the timestamp of each transaction, offering temporal information about when purchases occurred.

#### ItemCode: Provides a unique code for each item, enabling the distinction between different products in the dataset.

#### ItemDescription: Offers a brief description of the purchased item, enhancing the understanding of the nature of transactions.

#### NumberOfItemsPurchased: Quantifies the total number of items bought in each transaction, providing insights into the volume of goods purchased.

#### CostPerItem: Represents the cost associated with each individual item, contributing to the financial analysis of transactions.

#### Country: Specifies the country where the transaction occurred, providing geographical context for the market or location of purchases.

# B. Detailed description of the Product. You must describe why the design of the Product makes it Transactional.

Our product is an customer-transactional database designed to handle and record all transactions through any e-commerce platform. This database is structured to manage a vast amount of data that reflects the dynamic nature of online retail activities. It's created to serve as the backbone of an e-commerce business, ensuring the integrity, availability, and consistency of transaction data in real-time.

#### Why the Design Makes it Transactional:

**ACID Properties**:

The database is designed to adhere to the ACID (Atomicity, Consistency, Isolation, Durability) properties, which are essential for transactional systems. This means that each transaction is treated as a single unit of work, changes are made consistently, transactions do not interfere with each other, and once a transaction is committed, it will remain so, even in the event of a system failure.

#### Real-Time Processing:

Transactions are processed in real-time, ensuring that data such as inventory levels, customer orders, and financial records are always up to date. This immediate processing is crucial for an e- commerce platform where delays can result in poor customer experiences or financial discrepancies.

#### Concurrent Access:

#### The "Customers.csv" dataset and the "Transactional.csv" dataset are both structured for high concurrency, enabling numerous transactions to take place simultaneously without conflicts. This is of paramount importance for e-commerce websites that handle numerous customers and transactions concurrently.

#### Data Integrity and Security:

Ensuring data integrity involves maintaining the accuracy and consistency of data over its entire lifecycle. The database is equipped with constraints, triggers, and authorization controls to safeguard data integrity and enforce business rules. Security measures such as encryption and access control are also in place to protect sensitive transaction data against unauthorized access or breaches.

#### Scalability:

As the e-commerce business grows, the database can scale to handle increasing transaction volumes without degradation in performance. This means implementing scalable architectures like sharding or partitioning to distribute the load and optimize performance.

#### Logging and Audit Trails:

The database keeps logs and audit trails of all transactions, enabling the tracking of changes and supporting the auditability of the system. This is necessary for compliance with financial regulations and for resolving any disputes that may arise.

#### Robust Backup and Recovery:

To prevent transaction loss, the database includes robust backup and recovery mechanisms. It ensures that the system can quickly recover from hardware failures, power outages, or other unexpected issues without data loss.

#### Support for Complex Transactions:

The database supports complex transactions, which may include multiple steps or stages. For example, an e-commerce transaction could involve inventory checks, payment processing, and order confirmation, all of which need to be completed successfully for the transaction to be considered complete.

In conclusion, the transactional database is engineered specifically to manage, store, and secure transactions for online retail businesses. Its transactional nature is characterized by the ability to handle many quick, simultaneous transactions while maintaining data integrity, consistency, and reliability, which are all critical for the success of an e-commerce platform.

# A detailed description of the Product data structures.

A diagram of a customer transaction

Description automatically generated with medium confidence

### Tables:

Tables are database objects that store data in rows and columns and are the same elements we used and dealt with in mssql or sql server in class.

### Create Tables syntax:

CREATE TABLE table\_name (

column1 datatype, column2 datatype,

...

CONSTRAINT constraint\_name PRIMARY KEY (column\_name), CONSTRAINT constraint\_name UNIQUE (column\_name),

CONSTRAINT constraint\_name FOREIGN KEY (column\_name) REFERENCES reference\_table (reference\_column));

* ***table\_name:*** *The name of the table.*
* ***column1 datatype, column2 datatype, ...:*** *Define the table's columns and their data types.*
* ***CONSTRAINT constraint\_name PRIMARY KEY (column\_name):*** *Specifies a primary key constraint.*
* ***CONSTRAINT constraint\_name UNIQUE (column\_name):*** *Specifies a unique constraint.*
* ***CONSTRAINT constraint\_name FOREIGN KEY (column\_name) REFERENCES reference\_table (reference\_column):*** *Specifies a foreign key constraint.*

### Sequences:

A sequence is a database object that generates unique values. It is often used to generate unique primary key values. We had to create these to store our latest primary key value for any table we create as an equivalent to ‘Identity(1,1)’ function in oracle to generate our primary keys.

### Create sequence syntax:

CREATE SEQUENCE sequence\_name START WITH initial\_value INCREMENT BY increment\_value MAXVALUE max\_value

MINVALUE min\_value NOCACHE/NOCYCLE;

* ***sequence\_name:*** *The name of the sequence.*
* ***START WITH initial\_value:*** *The initial value of the sequence.*
* ***INCREMENT BY increment\_value:*** *The step size by which the sequence increases.*
* ***MAXVALUE max\_value:*** *The maximum value the sequence can reach.*
* ***MINVALUE min\_value:*** *The minimum value the sequence can reach.*
* ***NOCACHE:*** *This option disables caching of sequence values.*
* ***NOCYCLE:*** *This option prevents the sequence from cycling back to the minimum value after reaching the maximum value.*

### Triggers:

A trigger is a set of actions that are automatically executed when a specific database event occurs, such as an INSERT, UPDATE, or DELETE operation. Triggers helped us increment our primary key values upon insertion into the table finally creating the effect of the ‘identity(1,1)’ we discussed in class and is used in mysql.

### Create Triggers syntax:

CREATE OR REPLACE TRIGGER trigger\_name

BEFORE/AFTER INSERT/UPDATE/DELETE ON table\_name FOR EACH ROW

BEGIN

-- Trigger logic here END;

* ***trigger\_name:*** *The name of the trigger.*
* ***BEFORE/AFTER:*** *Specifies whether the trigger fires before or after the triggering event.*
* ***INSERT/UPDATE/DELETE:*** *Specifies the triggering event that activates the trigger.*
* ***table\_name:*** *The name of the table to which the trigger is associated.*
* ***FOR EACH ROW:*** *Indicates that the trigger operates on each affected row.*
* ***BEGIN...END;****: Contains the trigger's PL/SQL logic.*

In these examples, replace sequence\_name, trigger\_name, table\_name, column\_name, datatype, and other placeholders with your specific names and values.

# A detailed description of the CRUD operations if it is Transactional:

Create (INSERT) Operations:

CREATE TABLE CUSTOMER (

CustomerID int NOT NULL,

Gender varchar(255),

Age int NOT NULL,

Annual\_Income int NOT NULL,

Spending\_Score int NOT NULL,

Profession varchar(255),

Work\_Experience int,

Family\_Size int NOT NULL,

GenderID int NOT NULL,

ProfessionID int NOT NULL,

PRIMARY KEY (CustomerID)

);

-- Create sequences for generating unique values

CREATE SEQUENCE GenderID\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE ProfessionID\_seq START WITH 1 INCREMENT BY 1;

-- Create triggers to automatically generate values for GenderID and ProfessionID

CREATE OR REPLACE TRIGGER trg\_generate\_GenderID

BEFORE INSERT ON CUSTOMER

FOR EACH ROW

BEGIN

:new.GenderID := GenderID\_seq.nextval;

END;

/

CREATE OR REPLACE TRIGGER trg\_generate\_ProfessionID

BEFORE INSERT ON CUSTOMER

FOR EACH ROW

BEGIN

:new.ProfessionID := ProfessionID\_seq.nextval;

END;

/

Insert into CUSTOMER Table:

-- Inserting a single row into the CUSTOMER table

INSERT INTO CUSTOMER (CustomerID, Gender, Age, Annual\_Income, Spending\_Score, Profession, Work\_Experience, Family\_Size, GenderID, ProfessionID)

VALUES (1, 'Male', 30, 50000, 75, 'Engineer', 5, 3, 1, 1);

Inserts customer data into the "CUSTOMER" table. It creates new customer records with details such as CustomerID, Gender, Age, Annual Income, Spending Score, Profession, Work Experience, Family Size, GenderID, and ProfessionID.

Insert into CUSTOMER Table (Multiple Rows):

-- Inserting multiple rows into the CUSTOMER table

INSERT INTO CUSTOMER (CustomerID, Gender, Age, Annual\_Income, Spending\_Score, Profession, Work\_Experience, Family\_Size, GenderID, ProfessionID)

VALUES (2, 'Female', 28, 45000, 80, 'Doctor', 8, 4, 2, 2);

INSERT INTO CUSTOMER (CustomerID, Gender, Age, Annual\_Income, Spending\_Score, Profession, Work\_Experience, Family\_Size, GenderID, ProfessionID)

VALUES (4, 'Female', 25, 55000, 70, 'Lawyer', 7, 3, 2, 4);

INSERT INTO CUSTOMER (CustomerID, Gender, Age, Annual\_Income, Spending\_Score, Profession, Work\_Experience, Family\_Size, GenderID, ProfessionID)

VALUES (3, 'Male', 35, 60000, 60, 'Teacher', 6, 2, 1, 3);

Inserts multiple rows of customer data into the "CUSTOMER" table. Similar to the previous operation, this operation creates multiple customer records.

Read (SELECT) Operations:

UPDATE CUSTOMER

SET Gender = 'Female', Age = 32

WHERE CustomerID = 1;

UPDATE CUSTOMER

SET Annual\_Income = 48000, Family\_Size = 3

WHERE CustomerID = 2;

UPDATE CUSTOMER

SET Spending\_Score = 70, Work\_Experience = 9

WHERE Age > 30;

Select from CUSTOMER Table:

Retrieves and displays all data from the "CUSTOMER" table.

Select from CUSTOMERS Table (Nonexistent):

Attempts to select data from a table named "CUSTOMERS," but this table does not exist. Therefore, it will result in an error.

Select from transactiondata Table:

Retrieves and displays all data from the "transactiondata" table.

Select from TRANSACTION\_DATA Table:

Retrieves and displays all data from the "TRANSACTION\_DATA" table.

Update Operations:

Update CUSTOMER Table:

Modifies customer records in the "CUSTOMER" table, changing the Gender and Age for customers with CustomerID equal to 1.

Update CUSTOMER Table (Annual Income and Family Size):

Updates customer records in the "CUSTOMER" table, specifically modifying Annual Income and Family Size for customers with CustomerID equal to 2.

Update CUSTOMER Table (Spending Score and Work Experience):

Updates customer records in the "CUSTOMER" table, setting Spending Score and Work Experience for customers older than 30.

Delete Operations:

Delete from CUSTOMER Table:

DELETE FROM CUSTOMER

WHERE CustomerID = 1;

DELETE FROM CUSTOMER

WHERE CustomerID = 2;

DELETE FROM CUSTOMER

WHERE CustomerID = 3;

DELETE FROM CUSTOMER

WHERE CustomerID = 4;

SELECT \* FROM CUSTOMER;

SELECT \* FROM CUSTOMERS;

DROP TABLE CUSTOMERS;

Removes a specific customer record from the "CUSTOMER" table based on the CustomerID.

Delete from CUSTOMER Table (Multiple):

Repeatedly deletes customer records from the "CUSTOMER" table based on CustomerID for multiple customers.

Delete from CUSTOMERS Table (Nonexistent):

Attempts to delete data from a table named "CUSTOMERS," but this table does not exist. Therefore, it will result in an error.

Drop Tables:

Attempts to drop (delete) tables named "CUSTOMERS" and "transactiondata," which may result in the removal of these tables and their associated data if they exist.

Delete Table CUSTOMERS (Nonexistent):

DELETE TABLE CUSTOMERS;

Drop Table TRANSACTION\_DATA:

DROP TABLE transactiondata;

SELECT \* FROM transactiondata;

SELECT \* FROM TRANSACTION\_DATA

Attempts to drop (delete) the "TRANSACTION\_DATA" table and its associated data.