Data Collection and Modeling

Course 9 - Databases

Definition

“A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.” (Oracle)

“database, also called electronic database, any collection of data, or information, that is specially organized for rapid search and retrieval by a computer. Databases are structured to facilitate the storage, retrieval, modification, and deletion of data in conjunction with various data-processing operations. A database management system (DBMS) extracts information from the database in response to queries.” (Britannica)

DBMS

- Is a software system used to store, retrieve, and query data. It can be seen as an interface between users and databases that facilitates operations to create, read, update, and delete data in the database.
- Provides data security, data integrity, concurrency, and data administration processes.
- Provides a centralized view of data that can be accessed by multiple users from different locations. It can offer several views of a single database schema by limiting what a user can see and how that data can be seen.

DBMS

Components:

- Storage engine: used to interact with the file system at the OS level
- Data dictionary: metadata for all objects that exist in the database
- Access language: API for data access (usually SQL)
- Optimization engine: parses query and translates it into commands for data access
- Query processor: used for running the (optimized) query and returning results
- Lock manager: manages concurrent access
- Log manager: ensures the record of changes (log) is accurate and efficient
- Data utilities: statistics, backup, recovery, integrity, load, download, repair…

DB Types

- Relational
- Object oriented
- Distributed
- Data warehouses
- NoSQL
- Cloud
- Multimodel

Relational DBMS

- also known as SQL DBMS, it stores/represents data as rows in tables with a fixed schema and enforces relationships defined by values in key columns.
- adaptable to most use cases.
- simple data representation
- queries expressed using a simple high level language
- suitable also for non technical users

Relational Model

- based on relations: data collection is organized in tables (that represent relations)
- a relation has a schema and an instance
- the schema describes the relation name, the header (table columns/fields) and the domain for each field; the domain of a field has a name and a set of associated values
- the instance is the table itself

Relational Model

- the attribute values are atomic and scalar
- the records in the instance are distinct
- the records are not ordered !!!
- relations must be normalized to avoid certain anomalies

Relational Model

- Integrity constraints: a set of predefined rules that are used to ensure integrity, validity and consistency of data in the database table
- Rules are evaluated every time a data modification operation is applied (insert, update, delete or alter)
- Types of constraints:
  - Domain: restricts the values for a column
  - Entity integrity: ensures that there are no null primary keys
  - Referential integrity: maintains a valid relationship between 2 tables
  - Key: identify uniquely a row inside a table; there can be several keys, one being primary key (PK)

Relational Database

- Set of relations having distinct names
- Contains also schemas: schemas for the relations in the database
- There is a database instance: collection of relation instances that are valid by conforming with the integrity constraints

SQL DDL

Data Definition Language is a subset of SQL that is used to define the database schema. It is used to create, modify, and delete database objects such as tables, indexes, and views.

DDL statements are typically executed using a database management system, such as PostgreSQL, which provides both a command-line interface or graphical user interface for running statements.

DDL statements include CREATE, ALTER, and DROP. The CREATE statement is used to create new database objects, the ALTER statement is used to modify existing objects, and the DROP statement is used to delete objects.

SQL DDL

```
CREATE TABLE table_name(
column_definition
[, column_definition]
...
...
[, table_restrictions]
)
```

where

```
Column_definition = column_name data_type[(length)] [DEFAULT
value] [column_restriction]
```

SQL DDL

```
CREATE TABLE users (
    id INTEGER NOT NULL,
    username VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

SQL DDL

In SQL can be defined several restrictions; some of them are:

- NOT NULL - the column cannot have a NULL value
- UNIQUE - all values in the column are unique (one can be NULL)
- PRIMARY KEY - the combination of NOT NULL and UNIQUE. It is used to uniquely identify each row in a table
- FOREIGN KEY - Used to preserve links between tables
- CHECK - the values in the column satisfy the condition
- DEFAULT - Used as default value for the column if no value is specified

SQL DDL

```
CREATE TABLE users (
    id INTEGER NOT NULL,
    username VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);
```

SQL DDL

```
CREATE TABLE users (
    id INTEGER NOT NULL,
    username VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
        PRIMARY KEY (id)
);
```

SQL DDL

```
CREATE TABLE users (
    id INTEGER NOT NULL,
    username VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
     age int CHECK (age>=18),
        PRIMARY KEY (id)
);
```

SQL DDL

```
CREATE TABLE users (
    id INTEGER NOT NULL,
    username VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
     is_admin VARCHAR(3) DEFAULT ‘no’,
        PRIMARY KEY (id)
);
```

SQL DDL

```
CREATE TABLE users (
    id INTEGER NOT NULL,
    username VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
     star_date date,
     end_date date,
        PRIMARY KEY (id),
        CONSTRAINT check_dates (start_date < end_date)
);
```

SQL DDL

```
CREATE TABLE user_posts (
    id INTEGER NOT NULL,
    user_id INTEGER NOT NULL,
    post VARCHAR(255) NOT NULL,
    post_date date,
        PRIMARY KEY (id),
        FOREIGN KEY (user_id) REFERENCES users(id)
);
```

SQL DDL

`ALTER TABLE` is used to add, delete, modify columns in an existing table or to add and drop various constraints on an existing table.

```
ALTER TABLE table_name ADD column_name datatype;

ALTER TABLE users ADD city varchar(100);
```

```
ALTER TABLE table_name DROP COLUMN column_name;

ALTER TABLE users DROP COLUMN city;
```

SQL DDL

```
ALTER TABLE table_name ALTER|MODIFY COLUMN column_name datatype;

ALTER TABLE users ALTER COLUMN city TYPE varchar(150);


ALTER TABLE table_name ALTER COLUMN column_name [SET DEFAULT
value | DROP DEFAULT];

ALTER TABLE table_name ALTER COLUMN column_name [SET NOT NULL|
DROP NOT NULL];

ALTER TABLE table_name ADD CHECK expression;

ALTER TABLE table_name ADD CONSTRAINT constraint_name
constraint_definition;
```

SQL DDL

`DROP TABLE` is used to drop an existing table in a database.

```
DROP TABLE table_name;

DROP TABLE users;
```

`TRUNCATE TABLE` is used to delete the data inside a table, but not the table.

```
TRUNCATE TABLE table_name;

TRUNCATE TABLE users;
```

Exercise

Model the following entities:

- Customers: name, email, address
- Orders: order_date, products
- Products: name, description, category, price
- Categories: name, description

SQL DML

`INSERT INTO` is used to insert records in a table.

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);


INSERT INTO users (id, name, email)
VALUES (1, ’John Doe’, ’jon.doe@email.com’)
```

If the values are specified in the order the columns were created in the table, the list of columns can be omitted.

SQL DML

`INSERT INTO` is used to insert records in a table.

```
INSERT INTO table_name (column1, column2, column3, ...)

SUBQUERY;
```

Where `SUBQUERY` is a `SELECT` statement that generates a set of records.

```
INSERT INTO users (id, name, email)

SELECT id, name, email FROM other_users;
```

SQL DML

UPDATE is used to modify existing records in the table.

```
UPDATE table_name

SET column1 = value1, column2 = value2, ...

[WHERE condition];

UPDATE users

SET name = ‘Jane Doe’ WHERE id = 1;
```

SQL DML

UPDATE is used to modify existing records in the table.

`Condition`, if used, specifies which records in the table are changed; if omitted, all records in the table will have the specified columns changed to the specified value.

SQL DML

`DELETE` is used to remove existing records in the table.

```
DELETE FROM table_name [WHERE condition];
```

```
DELETE FROM users WHERE id = 1;
```

`Condition`, if used, specifies which records in the table are removed; if omitted, all records in the table will be erased.

SQL DML

Conditions are based on SQL filters; they are text strings that specify a set of comparisons for data items that are returned when condition holds true.

The syntax is:

```
Field  Operator      Value

[AND | OR | NOT (Field      Operator      Value) ...]
```

where `Field` is the name of a table field, `Operator` is a (comparative) operator, and `Value` is the field value.

SQL DML

Filter Conditions:

```
expression comparison_operator expression

expression [NOT] BETWEEN valmin AND valmax

expression [NOT] LIKE pattern ("%" any substring, "_" one character)

expression IS [NOT] NULL

expression [NOT] IN (value [, value] ...)

expression [NOT] IN (subquery)

expression comparison_operator {ALL | ANY} (

[NOT] EXISTS (subquery)
```