SQL Advanced

03 August 2022 21:46

Anomaly:

- Mismatch/Inconsistencies of data
- Types:
    - Insertion Anomaly
    - Deletion Anomaly
    - Updation Anomaly

Normalization:

- Decompose large,complex tables to simpler,smaller ones
- ***Functional Dependency***: If a column depends on a primary key
- ***Partial Dependency*** : If a column depends on a subset of primary key
- ***Transitive Dependency***: If a column depends on a non-primary key
- ***Types of Normal Forms:***
    1. ***First Normal Form(1NF):***All elements are atomic(single) in nature and no repeating elements
    2. ***Second Normal Form(2NF):*** It is in 1NF. No partial Dependency exists
    3. ***Third Normal Form(3NF):*** It is in 2NF. No partial Dependency exists

Subqueries and Query Expressions:

- Query within a main query that operates individually and shares result with a main query to reduce complexity
- **Types:**
    - ***Single Row Subquery***: Feeds single value to main query
    - ***Multiple Row Subquery***: Returns more number of rows
    - ***Multiple Column Subquery***:Returns one or more columns which matches outer column query
    - ***Correlated Subquery***: Subquery dependent on outer query for retrieving each record
    - ***Nested Subquery***: Subquery within an another subquery. This is called Nesting.
    - **Subqueries in WHERE clause:**
        - Used for single row comparision like =,>,<,>=,<=
        - Used for multi row comparision like EXIST,IN,ANY,ALL
            - **Subqueries in IN clause:**When main query searches all rows in returned by subquery.
            - **Subqueries in EXISTS clause:** When main query has one or more matching records resulted by subquery.
            - **Subquery Quantify Test:** Validating many records and checking if any one record matches of a main query matches with multiple records of subquery.
              It is done by ***ANY clause***: Use like > Any or < Any
    - **Subqueries in WITH clause:**
        - With clause serves the result of subquery whenever the main query references to it.
        - Wherever the With clause is executed, same subquery can be used multiple times without executing again. It is mostly used in complex queries,to avoid calling subqueries repeatedly.

- Subqueries are dynamic not static(fixed)

Analytical/Window Functions/Online Analytical Processing (OLAP)Functions:

- Syntax: Over clause to calculate aggregate results on a group of rows based on candidate key
    - fn() over(...) ,or,
    - fn() over() (If there is nothing to b applied over)
- Comparision between current record entry with aggregated result
- **Rank():** Skips rank if the previous rank is duplicated of candidate key *[Rank() Over()]*
- **Dense_Rank()**: Doesn't skips rank if the previous rank is duplicated of candidate key *[Dense_Rank() over()]*
- **Percent_Rank():**
- **Row_number():** Displays unique ID to identify rows irrespective of value *[Row_number() over()]*
- **LAG():** Compare values with previous row [lag(column,how many rows to be lagged)]
- **LEAD():** Compare values with next/following row [lead(column,how many rows to be leaded)]
- **First_Value():**Analyses and returns the first value from ordered set of rows from over()[First_Value() over()]

- **Last_Value():**Analyses and returns the last value from ordered set of rows from over()[Last_Value() over(… range between unbounded preceding and unbounded following)]
- **NTILE():**Splits the record into predefined number of buckets. [ntile(no. of buckets) over()]
- **CUME_DIST()**: Percentage of record occupied by total record[cume_dist() over()]
- Aggregate function within analytical function:
  - AVG() over()
  - MIN() over()
  - MAX() over()
  - SUM() over()
  - COUNT() over()

Data Integrity:

- Ensures smooth flow of business and ensures consistency of data through entire product cycle.
- Classified into :
  - **Row Level Integrity:** Each row in the table is referenced by unique values. Eg: Customer Id in customer table
  - **Column Level Integrity:** Ensures to insert correct values according to its datatype. Eg: Customer Id is an integer and accepts only integer.
  - **Referential Integrity**: Ensures consistency of records between two tables and establishes the relationship between two table sing FOREIGN KEY . Syntax: FOREIGN KEY (column name) references tablename (column name)
    - Foreign key doesn't allow to 'INSERT' or 'UPDATE' values in child table which is not in parent table also cant delete a row in parent table as the value of the row is existing or used in child table

Entity Constraints:

- Rules or limit of data that can be put into a column/name in SQL constraints.
- Most common constraints: NOT NULL,DEFAULT and others (UNIQUE,PRIMARY KEY,CHECK)
- Classified *into:*
  - **Column Level Constraint**: Limits only column level data values
  - **Table Level Constraint**: Limits complete table dataset
- **CHECK Constraint:** Checks the values before insertion of records and prevents unwanted entries.[Alter table employees add check (condition)]
- **Uniqueness Constraint:** Checks value are UNIQUE
- **PRIMARY KEY Constraint:** Unique for a column and will act as an index for the column
- ***Delete and Update rules:***
  - Foreign key always referenced to primary/unique key
  - If Foreign key is NULL, it is not referenced to any primary key of another table.
  - If Foreign key is assigned to any value, it is referenced to primary key
- ***Cascading Rules:***
  - To get rid of foreign key problem with Referential Integrity,so changes in parent table is automatically updated in child table
  - Can be done for two: UPDATE CASCADE,DELETE CASCADE as
    (….
    on DELETE CASCADE
    on UPDATE CASCADE); while creating a table

Virtual Table:

- Derived form of data from physical database table and are logically represented using physical data but do not store any data.
- Otherwise called as Views and are stored in mysql information_schema annd can be retrieved without rewriting again

View:

- An object in a database that can be created using SELECT query in combination with complex logic.
- Types of View:
  - Simple View: Select query is written using simple table. *Create VIEW viewname as SELECT * from tablename.*
  - Complex View: Select query with multiple tables like join,subquery,clauses or conditional filters
  - Inline View: Only used in FROM clause of SELECT query.defined dynamically in queries.
  - Horizontal View: Represents the data without aware of columns and it doesnot need to know the number of columns and datatypes.
  - Vertical View
  - Joined View
  - Drop View : Drops the view

Transaction Processing: (Before doing transaction processing in mysql, set autocommit=0) -- we are disabling the autocommit

- START TRANSACTION :To start transaction

- COMMIT : To save the transaction permanently
- ROLLBACK: To revert or rollback the transaction. *ROLLBACK TO SAVEPOINT_NAME;*
- SAVE POINT:Partial Revert back. *SAVEPOINT SAVEPOINT_NAME;*
- RELEASE POINT:When save point is no longer required in our query then we can use this.

ACID:

- A for Atomicity, C for Consistency,I for Isolation,D for Durability

Isolation: SET TRANSACTION ISOLATION LEVEL

- Isolating one user transaction from other by forming queue of transactions.
- By default the isolation command is repeatable-read.
- Set transaction sets the isolation levels by restricting other transactions perfoming tasks on the same records.Other transactions will wait in queue to perform.
- Different levels of isolation:
    - **READ UNCOMMITTED** : Lowest isolation level, Reading uncommitted data is known as Dirty Read.Transactions are not logically isolated from each other.Here the uncommitted data of first user, will be visible to the next user.

    - **READ COMMITTED** : Doesn't perform Dirty Read and only committed data is available to read by the next user.This transaction will acquire the read or write lock for current row.

    - **REPEATABLE - READ** : Default in mysql. Acquires read lock while reading and write locks while writing on all rows which are currently updated.The other user cannot read or write so this transaction non repeatable read. Suffers from Phantom Reads.

    - **SERIALIZABLE**: Highest isolation level and provides complete isolation effects to the transaction from other transaction.

Locking:

- Acquired by user transaction in a session
- SQL Engine locks the transaction when the transaction begins the DML statements
- After transaction, LOCKs are released.
- Locks Hierarchy : Database -> Table ->Page ->Row level
- Types of locks in Row level:
    - ***Shared Lock or Read Lock*** :Allows the user to read the data and not to write and locks for other users. Avoids deadlocks(avoids other users to write at the time we are viewing).Can be acquired in SELECT statement.
      [select * from sample LOCK IN SHARE MODE;]

    - ***Exclusive (x) Lock or Write Lock*** :Allows the user to write the data and locks for other users.Can be acquired in DELETE,INSERT,UPDATE statement.Only the user can access the data not other users while modifying/writing the data.
      [select * from sample for UPDATE;]

After the DML operations has been made and COMMIT happens, the other users are able to access.