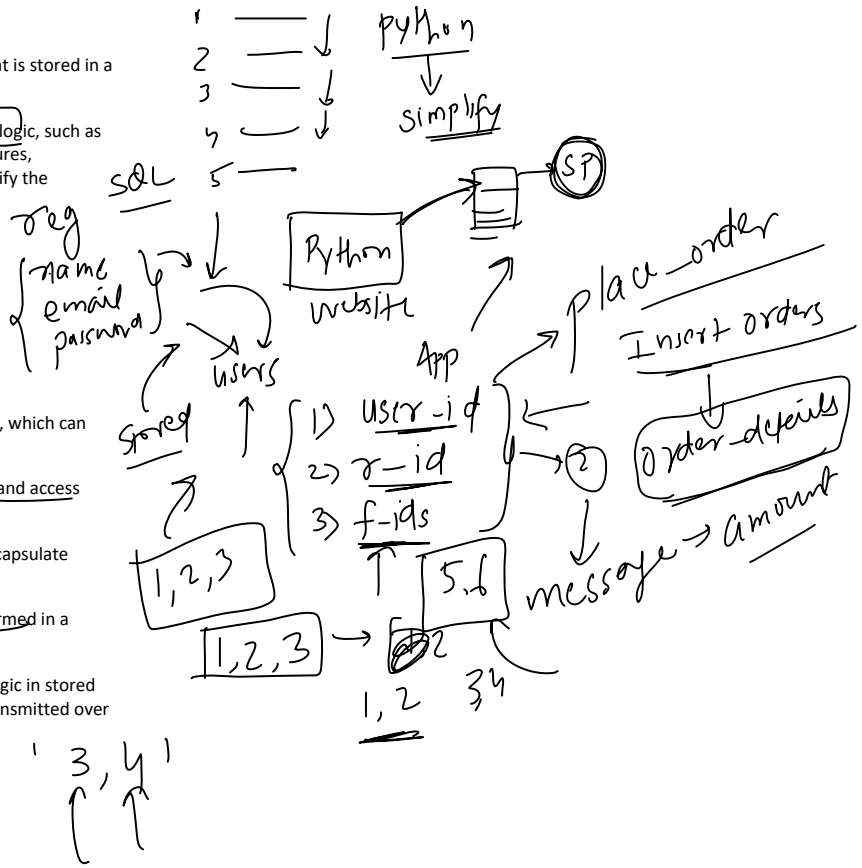
Stored Procedures

25 March 2023 16:39

A stored procedure is a named block of SQL statements and procedural logic that is stored in a database and can be executed by a user or application.

Stored procedures are often used to encapsulate business logic and application logic, such as data validation, data processing, and database updates. By using stored procedures, developers can separate application logic from the presentation layer and simplify the application code.

- Create hello world stored procedure
- Create stored procedure to create a new user
- Show error message
- Create stored procedure to show orders placed by 1 single user
- Create a stored procedure to place an order



Some of the benefits of using stored procedures include:

1. **Improved performance**: Stored procedures are precompiled and optimized, which can improve performance and reduce network traffic.
2. **Enhanced security**: Stored procedures can be granted specific permissions and access rights, which can improve security and limit access to sensitive data.
3. **Encapsulation of business logic**: Stored procedures allow developers to encapsulate complex business logic and make it easier to maintain and update.
4. **Consistency**: Stored procedures ensure that database operations are performed in a consistent manner, which can help to maintain data integrity.
5. **Reduced network traffic**: By encapsulating data access and manipulation logic in stored procedures, developers can reduce the amount of data that needs to be transmitted over the network.

Transactions

25 March 2023 08:19

What are Transactions?

A database transaction is a sequence of operations that are performed as a single logical unit of work in a database management system (DBMS). A transaction may consist of one or more database operations, such as inserts, updates, or deletes, which are treated as a single atomic operation by the DBMS.

It follows the principle of all or none.

What is Commit, Rollback and Savepoint?

In a database transaction, there are three main commands that are used to manage the transaction:

1. **Commit**: A commit command is used to permanently save the changes made by a transaction to the database. When a transaction is committed, all changes made by the transaction are made permanent and cannot be rolled back.
2. **Rollback**: A rollback command is used to undo the changes made by a transaction and return the database to its state before the transaction began. When a transaction is rolled back, all changes made by the transaction are discarded and the database is returned to its previous state.
3. **Savepoint**: A savepoint command is used to mark a specific point within a transaction where a rollback can be performed. This allows for partial rollbacks of a transaction, where only changes made after the savepoint are undone, while changes made before the savepoint are still committed to the database.

What is Autocommit?

Autocommit is a feature of database management systems (DBMS) that automatically commits each individual database transaction as soon as it is completed, rather than requiring an explicit commit command to be issued.

When Autocommit is enabled, each individual SQL statement issued against the database is treated as a separate transaction and is committed immediately after it is executed. This means that each SQL statement becomes a separate, independent transaction, and its effects are immediately visible to other users.

- Each SQL write statement is Autocommit (prove)
- set Autocommit = 0 and (show)
- START TRANSACTION -> show operations without committing
- START TRANSACTION -> with commit
- START TRANSACTION -> all or none with commit
- rollback
- rollback with savepoint
- rollback and commit together

What is ACID properties of a Transaction?

ACID is an acronym that stands for Atomicity, Consistency, Isolation, and Durability, which are a set of properties that ensure reliable database transactions:

1. **Atomicity**: This property ensures that a transaction is treated as a single, indivisible unit of work. This means that either all of the changes made by a transaction are committed to the database, or none of them are. If any part of the transaction fails, the entire transaction is rolled back, and all changes are undone.
2. **Consistency**: This property ensures that a transaction takes the database from one valid state to another valid state. It requires that all data in the database must conform to a set of rules, or constraints, which ensure data integrity.
3. **Isolation**: This property ensures that concurrent transactions do not interfere with each other. It requires that each transaction executes as if it were the only transaction executing against the database, even if multiple transactions are executing at the same time.
4. **Durability**: This property ensures that once a transaction is committed, its changes are permanently stored in the database, even in the event of a system failure or power outage. This is typically achieved through the use of database backups, replication, or other forms of data redundancy.

Together, these properties ensure that database transactions are reliable, consistent, and accurate, and that the data stored in a database is both protected and available at all times. The ACID properties are essential for mission-critical applications that require high levels of data integrity and availability, such as banking, finance, and healthcare systems.