Investigate Database Architecture Issues

Hardik Nahata, Mansi Pravin Thanki, Bhoomika Jethwani

Q1: What are cursors? Why are they useful in application architectures? What are some of the benefits? Are there drawbacks to cursors? How do MySQL and SQLite support cursors and how would you (in one of them) use them? Provide a 300-500 word explanation of cursors.

Answer:

What are cursors?

- When working with SQL, a dedicated memory is created by Oracle for processing the queries entered by the user.
- This dedicated memory is referred to as the Context Area.
- This context area contains all the information about the query, for example, the number of rows affected by the query.
- The **Cursor** is a pointer to the dedicated memory.
- A cursor stores the affected rows or resultant rows of a given SQL query.
- Cursor allows SQL to control the context area.
- The rows held by a cursor are called as the **Active Set**.
- Cursor also has a couple types, namely Implicit and Explicit cursors.
- Implicit Cursors are the ones which are created automatically when an SQL statement is triggered. They cannot be controlled and modified by developers.
- Explicit Cursors are used to gain better access or control to the context area. There cursors can be controller by programmers. They need to be declared in the declaration block of an SQL statement.

Why are they useful in application architectures?

- A DBMS, particularly a relational DB usually returns a set of rows on running a query, which satisfy the given condition or criteria of the query.
- Applications cannot work efficiently wiht the entire resultant set of rows of a query, and hence require a feature of mechanism which allows the application to process a one row or a small chunk of rows at a time.
- Cursors provide this exact functionality and hence are really useful in application architectures.

What are some of the benefits?

- One of the most useful feature of a cursor is that it allows a user to access a subset of rows from a query result instead of the complete set of rows. This is a very efficient way to glance at some results.
- It allows the user or client to update tables in a database while the cursor is in the process of getting the result for a query.
- Cursors are memory efficient for both the server and the client as they don't have to shell out big memory chunks.
- Cursors are ideal to be used on databases and tables which are busy and have multiple concurrent reads and writes ongoing.

Are there drawbacks to cursors?

- Cursors are network heavy. Basically every time a client executes an SQL query to fetch rows from a table, the cursor does a complete round trip from client to server using more resources.
- Each time a cursor is run, it takes much more bandwidth compared to single SQL statement which contains a simple SELECT or DELETE statement which makes only a single round trip of the network.
- Cursors point to the context area, which is a temporary storage are created in the system memory, which is also shared with other processes running in the system. This leads to usage of more resources.
- Cursors reduce performance and speed of the system.

How do MySQL and SQLite support cursors and how would you (in one of them) use them?

Support for cursors in MySQL

- Cursors in MySQL provide Read-Only mechanism, i.e. a user or client cannot alter, modify or update a table through a cursor.
- Cursors in MySQL are asensitive - these cursors create a temporary copy of the data returned from triggering a query. So it is important to know that updating a table on which a cursor is running can lead to mismatch in resultant data.

Support for cursors in SQLite

- Cursors are not supported in SQLite tradionally.
- Other flavors of SQLite such as sqlite3 in Python does support cursors.

How to use cursors in MySQL

- Cursors can be declared by using the DECLARE statement.
- Below is the syntax for declaring a cursor in MySQL: "DECLARE cursor_name CURSOR FOR SELECT statement;"
- To retrieve the results of a cursor, use the OPEN statement.
- Below is the syntax for opening a cursor in MySQL: "OPEN cursor_name;"
- A cursor can be deactivated with the CLOSE statement.
- Below is the syntax for closing a cursor in MySQL: "CLOSE cursor_name;"

References:

[1] https://www.redbooks.ibm.com/redbooks/pdfs/sg248326.pdf

[2] https://www.numerade.com/ask/question/topic-databse-what-are-cursors-why-are-they-useful-in-application-architectures-what-are-some-of-the-benefits-are-there-drawbacks-to-cursors-how-do-mysql-and-sqlite-support-cursors-and-how-w-58614/

[3] https://www.webcodeexpert.com/2013/11/what-is-cursor-advantages-and.html#:~:text=Cursors%20can%20provide%20t

Q2: What are connection pools? Why are they useful in application architectures? What are some of the benefits? Are there any potential drawbacks? What are some of the main issues with using connection pools? –> Answer:

What are connection pools?

1. **Connection:** Inorder to perform operations like insertion, update, deletion of data etc, an application needs to be connected to the database.
2. As the name suggests, **connection pool is a pool of open connections OR a cache of already opened database connections so that it can be reutilized by any requests that arrive in future instead of creating a new open connection for every request.**

3. If connection created -> place it in pool -> reutilized by different requests

   If all connections used -> create new connection and place it in pool

Why are they useful in application architectures?

1. Whenever an app scales bigger in size, the number of database related operations also increases in number.
2. It becomes very costly and time-inefficient to create a new connection every time for a new request.
3. Having a connection pool allows to reuse the created connections. This reduces time overheads, leads to faster execution and provides better user experience.
4. This also enables in improved resource management and reduces the cost for creating new connections.

What are some of the benefits?

1. Improve Performance:

Reutilizing connections in the pool reduces the time overhead that arises due to forming new connections every time a new request appears. This gives improved user experience due to faster execution.

2. Connection Reuse:

Connection pool reuses the connections in the pool rather than creating a new connection for every request. This saves resources and time as there is no additional time taken for creating new connections.

3. Improves Resource Management: - Since the connections inside the connection pool are reutilized, it reduces the burden of creating new connections and it improves management of the reduced number of resources.

4. Faster Connection time: - Since the connections are already open and active, the further requests reutilizes them and gives faster performance and improved user experience.

Are there any potential drawbacks?

1. Yes, if the pool size is quite small, the database operations will have to wait until the time a new active connetion is available. This introduces latency.
2. Additionally, connections can end up becoming stale in the connection pool.

What are some of the main issues with using connection pools?

1. Stale Connections:

One of the main issues of connection pools is that the connections may turn stale.

2. Connection Leakage:

In a scenario where a connection in connection pool is not closed, the connection pool will consider it as an active connection even though the connection is not being utilized.

3. Pool Fragmentation:

Pool Fragmentation is a problem that occurs when multiple large connection pools have been created by applications and they do not get freed up until that particular process exits. Since a large number of connections are open, a lot of memory gets used up and it results into reduced performance.

References:

[1] https://www.cockroachlabs.com/blog/what-is-connection-pooling/

[2] https://docs.oracle.com/en/database/oracle/oracle-database/21/jjucp/intro.html#GUID-C9CD25A8-67F5-4C27-8E48-BC21C8BE8D83

[3] https://medium.com/@digi0ps/connection-pooling-what-and-why-8d659e1530f9

[4] https://plumbr.io/blog/io/acquiring-jdbc-connections-what-can-possibly-go-wrong

[5] https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql-server-connection-pooling

Q3: REVIEW ON SQL INJECTION ATTACKS

Answer:

Some of the main issues addressed in the papers are: Brief on the SQL Injection Attacks. Threat that these attacks pose to various institutions and systems. Research on the effects and defense mechanisms of such attacks. Detection and Prevention techniques against these attacks. Strengths and weaknesses of these techniques.

What are SQL injection attacks(SQLIAs)? SQL injections are found to be one of the popular type of web attack technique which is used by attackers to add, update, remove information from sensitive data driven organizations. These type of attacks can take place on systems that are online or office and they may be web based or non web based. The SQL Injection is basically a code injecting technique that most hackers use to insert malicious SQL queries into the input fields in the underlying database. This technique is highly likely to take place in situations where there is unsuitable/incorrect code in web applications. When the input fields are made available directly for the user's input it can allow suck false SQL statements to go through as queries to the database.

Effects of SQLIAs on the system These attacks can cause a huge security risk as they cause breaches in the system's data. Moreover, they also make the hardware and the software components like local Servers and Web-Services, OS, Application software, Database Engines, etc more vulnerable. Hence it is most important to keep these components up to date with the latest security patches and updates so that they are less prone to such attacks.

Reasons for the SQLIA's There are various reasons/goals for the attacker to perform these kinds of attacks on the application systems. Below are some of the popular SQLIA's affecting systems in the modern databases: - Identifying injectable parameters - Performing database finger-printing - Determining database schema - Extracting data - Adding or modifying data - Evading detection - Bypassing authentication - Executing remote commands - Performing privilege escalation

Protection of Website and Applications against SQLIA's There have been several advancements and improvements in technology that can help to prevent or defend against these attacks. From the developers prospective, they can help to prevent such attacks by making use of various parameterized databases management queries with bounds and type arguments or with the use of parameterized stored procedures. Several firms use operations like monitoring, logging, validation, intrusion detection to update their system architectures in order to increase the security of the database.

Various Types of SQLIA: There are many different types of SQLIA attacks being performed which are mostly done together to solve a major purpose rather than used in isolation. Following are the various types as seen in the papers:

Tautologies- The main intent of this type of attack is to bypass the authentication in order to extract certain data from the system. The number of results returned in the query from this attacks defines the extent of damage caused. Attacker usually uses the WHERE conditional query for this attack.

Illegal/Logically Incorrect Queries - The main intent is to identify parameters to perform database finger printing. It allows the attacker to gain crucial information about the architectural structure of the back end of the web application.

Union Query- The main intent of this type of attack is to bypass the authentication in order to extract certain data from the system. In this the attacker uses the UNION SELECT query to exploit the vulnerable parameter changes in the data results returned. The attacker can trick the application to return data from other tables than the one specified by the coder.

Stored Procedures - These attacks aim to perform the stored procedures in the database. Once the attacker can figure out the database being used in the backend they can alter and execute the procedures according to their needs.

Piggy-Backed Queries - The main intent of the attacker in this case is to try and inject extra queries than those specified by the developer in the original query. This is different from the other types because these attacks do not try to modify the query, they instead try to add new and distinct queries that piggy back the original one.

Inference - The attacker in this case modifies the query to recast it in the form of an action that can be performed while answering atrue/ false question about the values for an entity in the database. They inject the sites so that it gets secured enough to give no usable feedback via the error message of the database.

Alternate Encodings - The attacker in this type of attack usually injects text that is modified so that they can avoid the detection by the defensive coding practices and the various automated prevention techniques used by the modern systems.There is no new and different way of this attack, they just enable the technique that allows the attackers to evade detection or prevention so as to exploit the system vulnerabilities. SQLIAS PREVENTION TECHNIQUES:

A. Defensive Coding Practices:

1. Positive Validation:

Instead of the negative pattern input validation, developers can check for valid inputs.

2. Checking input type:

Attacks can be avoided by simply checking the input type. For eg: rejecting numeric values where input is supposed to be characters

3. Encoding Inputs:

Encoding inputs involve that the meta-characters in a string are encoded in a fashion that database considers it as normal characters

B. Detection and Prevention Techniques:

1. Black Box Testing
2. Intrusion Detection Systems
3. Static Code Checkers
4. Combined Static and Dynamic Analysis

The three things that we learned or three key take-aways from the papers:

1. SQL Injection attacks and their types
2. Intent or reasons for attackers to perform these types of attacks
3. Ways of prevention / detection of these attacks.

Are there statements that you do not agree with?

No, there are no such statements in particular.

References:

[1] Atoum, J. O., & Qaralleh, A. J. (2014). A hybrid technique for SQL injection attacks detection and prevention. International Journal of Database Management Systems, 6(1), 21.

[2] https://northeastern.instructure.com/courses/110675/files/15521440?wrap=1