Concurrency issues

Last Updated: 2023-03-17

Because many users access and change data in a relational database, the database manager must allow users to make these changes while ensuring that data integrity is preserved.

Concurrency refers to the sharing of resources by multiple interactive users or application programs at the same time. The database manager controls this access to prevent undesirable effects, such as:

- **Lost updates**. Two applications, A and B, might both read the same row and calculate new values for one of the columns based on the data that these applications read. If A updates the row and then B also updates the row, A's update lost.
- **Access to uncommitted data**. Application A might update a value, and B might read that value before it is committed. Then, if A backs out of that update, the calculations performed by B might be based on invalid data.
- **Non-repeatable reads**. Application A might read a row before processing other requests. In the meantime, B modifies or deletes the row and commits the change. Later, if A attempts to read the original row again, it sees the modified row or discovers that the original row has been deleted.
- **Phantom reads**. Application A might execute a query that reads a set of rows based on some search criterion. Application B inserts new data or updates existing data that would satisfy application A's query. Application A executes its query again, within the same unit of work, and some additional (“phantom”) values are returned.

Concurrency is not an issue for global temporary tables, because they are available only to the application that declares or creates them.

Concurrency control in federated database systems

A *federated database system* supports applications and users submitting SQL statements that reference two or more database management systems (DBMSs) in a single statement. To reference such data sources (each consisting of a DBMS and data), the Db2® server uses nicknames. *Nicknames* are aliases for objects in other DBMSs. In a federated system, the Db2 server relies on the concurrency control protocols of the database manager that hosts the requested data.

A Db2 federated system provides *location transparency* for database objects. For example, if information about tables and views is moved, references to that information (through nicknames) can be updated without changing the applications that request this information. When an application accesses data through nicknames, the Db2 server relies on concurrency control protocols at the data source to ensure that isolation levels are enforced. Although the Db2 server tries to match the isolation level that is requested at the data source with a logical equivalent, results can vary, depending on data source capabilities.

– **Isolation levels**
The *isolation level* that is associated with an application process determines the degree to which the data that is being accessed by that process is locked or isolate from other concurrently executing processes. The isolation level is in effect for the duration of a unit of work.

– **Specifying the isolation level**
Because the isolation level determines how data is isolated from other processes while the data is being accessed, you should select an isolation level that balances the requirements of concurrency and data integrity.

– **Currently committed semantics**
Under *currently committed* semantics, only committed data is returned to readers. However, readers do not wait for writers to release row locks. Instead, readers retu

data that is based on the currently committed version of data: that is, the version of the data before the start of the write operation.

– **Option to disregard uncommitted insertions**
The **DB2_SKIPINSERTED** registry variable controls whether or not uncommitted data insertions can be ignored for statements that use the cursor stability (CS) or th read stability (RS) isolation level.

– **Evaluate uncommitted data through lock deferral**
To improve concurrency, the database manager in some situations permits the deferral of row locks for CS or RS isolation scans until a row is known to satisfy the predicates of a query.

Parent topic:

→ Application design