Db2

Change version

11.5

Show full table of contents

IBM Db2 Version 11.5 documentation
Product overviews
Database fundamentals
Database administration
Developing code for accessing and managing data
Database reference
Db2 Connect overview
Db2 containerized deployments
IBM Cognos Analytics 11.1
Federation
InfoSphere Data Replication
Spatial data
Glossary

Other resources on this topic

DB2 11 BLU Acceleration Implementation and Use
Course
DB2 11 for z/OS Application Performance and Tuning
Course
DB2 11 for zOS Implementation Workshop
Course
DB2 11 for zOS System Administration
Course
New Functions and Features in DB2 11 for zOS
Course
DB2 11 BLU Acceleration Implementation and Use
Course
DB2 11 BLU Acceleration Implementation and Use - SPVC
Course

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 return 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. 7.4 4)

Lock timeouts and deadlocks can occur under the cursor stability (CS) isolation level with row-level locking, especially with applications that are not designed to prevent such problems. Some high-throughput database applications cannot tolerate waiting on locks that are issued during transaction processing. Also, some applications cannot tolerate processing uncommitted data but still require non-blocking behavior for read transactions.

Currently committed semantics are turned on by default for new databases. You do not have to make application changes to take advantage of the new behavior. To override the default behavior, Set the **`cur_commit`** database configuration parameter to `DISABLED`. Overriding the behavior might be useful, for example, if applications require the blocking of writers to synchronize internal logic. During database upgrade from V9.5 or earlier, the **`cur_commit`** configuration parameter is set to `DISABLED` to maintain the same behavior as in previous releases. If you want to use currently committed on cursor stability scans, you need to set the **`cur_commit`** configuration parameter to `ON` after the upgrade.

Currently committed semantics apply only to read-only scans that do not involve catalog tables and internal scans that are used to evaluate or enforce constraints. Because currently committed semantics are decided at the scan level, the access plan of a writer might include currently committed scans. For example, the scan for a read-only subquery can involve currently committed semantics.

Because currently committed semantics obey isolation level semantics, applications running under currently committed semantics continue to respect isolation levels.

Currently committed semantics require increased log space for writers. Additional space is required for logging the first update of a data row during a transaction. This data is required for retrieving the currently committed image of the row. Depending on the workload, this can have an insignificant or measurable impact on the total log space used. The requirement for additional log space does not apply when **`cur_commit`** database configuration parameter is set to `DISABLED`.

Applications running under currently committed semantics will always disregard uncommitted insertions. See Option to disregard uncommitted insertions or ../../com.ibm.db2.luw.admin.regvars.doc/doc/r0005665.html for details.

Restrictions

The following restrictions apply to currently committed semantics:

- The target table object in a section that is to be used for data update or deletion operations does not use currently committed semantics. Rows that are to be modified must be lock protected to ensure that they do not change after they have satisfied any query predicates that are part of the update operation.
- A transaction that makes an uncommitted modification to a row forces the currently committed reader to access appropriate log records to determine the currently committed version of the row. Although log records that are no longer in the log buffer can be physically read, currently committed semantics do not support the retrieval of log files from the log archive. This affects only databases that you configure to use infinite logging.
- The following scans do not use currently committed semantics:
  - Catalog table scans. Currently committed semantics to apply only to read-only scans that do not involve internal scans that are used to evaluate or enforce constraints. Currently Committed does not apply to internal scans on catalog tables, but may be applied to external scans on catalog tables if the registry variable, **`DB2COMPOPT`**, is set to `LOCKAVOID_EXT_CATSCANS`.
  - Scans that are used to enforce referential integrity constraints
  - Scans that reference LONG VARCHAR or LONG VARGRAPHIC columns
  - Range-clustered table (RCT) scans
  - Scans that use spatial or extended indexes
- If an indoubt transaction occurs (due to an application crash or an unexpected transaction or resource manager outage), any rows that were locked by this indoubt transaction will remain locked until the indoubt transaction is resolved. During this time, a currently committed reader can read the currently committed version of these rows only if the indoubt transaction had previously used currently committed semantics prior to the crash or outage. If not, a currently committed reader will wait for the indoubt transaction to be resolved (and these locks to be released) before reading these rows.
- In a Db2®pureScale® environment, currently committed semantics apply to applications on any member. If a row reader is attempting to access a row which is being updated or deleted by an application on a remote member, the local member will retrieve the currently committed row data from the remote member. However, with the following restrictions:
  - If the remote member is down or performing member crash recovery, then currently committed semantics do not apply. The row reader will wait for member crash recovery to complete on the remote member and for the lock to be released, before reading the row.
  - If communication to the remote member fails while attempting to retrieve the currently committed row data, then currently committed semantics do not apply. The row reader will wait for the lock to be released before reading the row.
  - If the transaction which is updating or deleting the row on the remote member has committed or rolled back before the row

reader’s communication reaches it, then the row reader will attempt to read the row again. If a transaction on another remote member updates or deletes the row during this time, then the row reader will again attempt to retrieve the currently committed row data from that member. The row reader may give up after a number of attempts and instead wait for the lock to be released before reading the row data.

Monitoring

Currently committed row data retrievals can be monitored on a per-table basis through the **db2pd -tcbstats** option. See *CCLogReads*, *CCRemoteReqs*, *CCLockWaits*, and *CCRemRetryLckWs* values in ../../com.ibm.db2.luw.admin.cmd.doc/doc/r0011729.html.

Examples

Example 1:
Consider the following scenario, in which deadlocks are avoided by using currently committed semantics. In this scenario, two applications update two separate tables, as shown in step 1, but do not yet commit. Each application then attempts to use a read-only cursor to read from the table that the other application updated, as shown in step 2. These applications are running under the CS isolation level.

Step	Application A	Application B
1	update T1 set col1 = ? where col2 = ?	update T2 set col1 = ? where col2 = ?
2	select col1, col3, col4 from T2 where col2 >= ?	select col1, col5, from T1 where col5 = ? and col2 = ?
3	commit	commit

Without currently committed semantics, these applications running under the cursor stability isolation level might create a deadlock, causing one of the applications to fail. This happens when each application must read data that is being updated by the other application.
Under currently committed semantics, if one of the applications that is running a query in step 2 requires the data that is being updated by the other application, the first application does not wait for the lock to be released. As a result, a deadlock is impossible. The first application locates and uses the previously committed version of the data instead.

Example 2:
Consider the following scenario, in a Db2®pureScale® environment, in which an application avoids a lock wait condition. Application-A on member 1 has updated data on table T1 but not yet committed its changes, and application-B on either member 1 (same member as Application-A) or member 2 (different member then Application-B) using cursor stability isolation level attempts to read that data:

Step	Application-A on pureScale member 1	Application-B on any pureScale member
1	update T1 set col1 = 12where col2 = ‘Ava’	
2		select col1 from T1 where col2 = ‘Ava’
3	commit	

Without currently committed semantics, application-B would wait until application-A committed its update and released the row lock, before reading the data. Under currently committed semantics, application-B will use the previously committed version of the data instead.

Parent topic:

→ Concurrency issues

Related reference

→ cur_commit - Currently committed configuration parameter

→ LIST INDOUBT TRANSACTIONS command

Tell us what you think

Was this topic helpful?

Yes No