Improving dynamic SQL performance by enabling the dynamic statement cache

The *dynamic statement cache* is a pool in which Db2 saves control structures for prepared SQL statements that can be shared among different threads, plans, and packages. By sharing these control structures, applications can avoid unnecessary preparation processes and thus improve performance.

About this task

Introductory concepts

Submitting SQL statements to Db2

Dynamic SQL applications

Embedded dynamic SQL

As the Db2 ability to optimize SQL has improved, the cost of preparing a dynamic SQL statement has grown. Applications that use dynamic SQL might be forced to pay this cost more than once. When an application performs a commit operation, it must issue another PREPARE statement if that SQL statement is to be executed again. For a SELECT statement, the ability to declare a cursor WITH HOLD provides some relief but requires that the cursor be open at the commit point. WITH HOLD also causes some locks to be held for any objects that the prepared statement is dependent on. Also, WITH HOLD offers no relief for SQL statements that are not SELECT statements.

6.6.2

Db2 can save prepared dynamic statements in a cache. The cache is a dynamic statement cache pool that all application processes can use to save and retrieve prepared dynamic statements. After an SQL statement has been prepared and is automatically saved in the cache, subsequent prepare requests for that same SQL statement can avoid the costly preparation process by using the statement that is in the cache. Statements that are saved in the cache can be shared among different threads, plans, or packages.

For example, assume that your application program contains a dynamic SQL statement, STMT1, which is prepared and executed multiple times. If you are using the dynamic statement cache when STMT1 is prepared for the first time, it is placed in the cache. When your application program encounters the identical PREPARE statement for STMT1, Db2 uses the already prepared STMT1 that is saved in the dynamic statement cache. The following example shows the identical STMT1 that might appear in your application program:

```
PREPARE STMT1 FROM ...     Statement is prepared and the prepared
EXECUTE STMT1              statement is put in the cache.
COMMIT
⋮
PREPARE STMT1 FROM ...     Identical statement. DB2 uses the prepared
EXECUTE STMT1              statement from the cache.
COMMIT
⋮
```

You must enable the dynamic statement cache before it can be used.

Procedure

To enable the dynamic statement cache to save prepared statements:

Specify YES for the value of the CACHEDYN subsystem parameter.

- **Dynamic SQL statements that Db2 can cache**
  Only certain dynamic SQL statements can be saved in the dynamic statement cache.
- **Conditions for statement sharing**
  If a prepared version of an identical SQL statement already exists in the dynamic statement cache, certain conditions must still be met before Db2 can reuse that prepared statement.
- **Capturing performance information for dynamic SQL statements**
  Db2 maintains statement caching performance statistics records when dynamic statements are cached. The statistics include cache hit ratio and other useful data points that you can use to evaluate the overall performance of your statement caches and statement executions.
- **Invalidation of cached dynamic statements**
  Various actions and events can invalidate statements in the dynamic statement cache. Db2 uses the full prepare process to generate new access paths for invalid cached dynamic SQL statements.
- **Invalidating statements in the dynamic statement cache**
  |< You can control whether Db2 invalidates statements in the dynamic statement cache when you collect statistics for objects that are referenced in the statements. In a data sharing environment, the relevant statements are also invalidated in the cache of other members in the group. Statistics collection does not invalidate cached statements by default. >|

Parent topic:

→ Improving dynamic SQL performance

Related concepts

→ Methods for keeping prepared statements after the point of commit or rollback

→ Dynamic SQL applications

Related tasks

→ Including dynamic SQL in your program

→ Capturing performance information for dynamic SQL statements

→ Monitoring the dynamic statement cache with READS calls

→ Calculating the EDM statement cache hit ratio

→ Enabling dynamic SQL statement caching for ODBC function calls

Related reference

→ CACHE DYNAMIC SQL field (CACHEDYN subsystem parameter)

Related information

Dynamic Statement Cache (white paper)