Over the past few years, much has been written about DB2 design and implementation best practices. There are many books, along with dozens of articles, that provide insight into how DB2 should be designed and integrated into applications. To provide some basic guidance for customers and business partners entering this maze, this article compiles a list of arguably the best practices to be considered during design and implementation of your application using IBM® DB2® technology. The intended result of this article is a more effective and efficient DB2 implementation. This article is not intended to provide detailed performance best practices for existing DB2 customers and independent software vendors (ISVs).

Introduction

DB2 for Linux, UNIX and Windows provides a database platform on which robust, on-demand business applications can be built. DB2 provides optimized management of relational data, and with DB2 9, optimized management of XML data as well.

This article assumes that you have a basic knowledge of relational database systems (RDBMS) and have previously implemented applications using either DB2 or another RDBMS. Common best practices are provided that assist you in basic design for performance and in creating an RDBMS environment that are easy to operate and maintain. As such, this article leaves out whole areas of database design, such as security, backup/recovery, and failover. The article also highlights some common tips in other areas, such as performance, without going into an extensive discussion of database analysis and design issues for these topic areas. Some additional resources are mentioned that will hopefully provide you with more extensive information on all DB2 topic areas. Hopefully, the common best practices listed in this article will assist those responsible for application deployments by reducing the effort needed to deploy, manage, and improve the performance of DB2 related applications.

General advice

There is a vast amount of DB2 technical material that can be utilized to assist those responsible for implementing applications with DB2 for Linux, UNIX, and Windows. Sorting through the
various sources of information can be daunting for both novice and experienced practitioners. The following Web sites are recommended as starting points for DB2 technical material available on the Internet.

The **Information Management developerWorks zone** that contains this article also contains many other DB2 articles, a wide array of technical materials, demo code, and product trials for download. The **Information Management developerWorks Technical Library** contains a wide range of content, including technical articles, product documentation, and tutorials.

There is an abundance of written materials on DB2. One of the best sources for DB2 related publications is the **IBM Press**. In particular the following books may be of interest:

- "**Understand DB2: Learning Visually with Examples.**" The **2nd edition** of this book has been written for DB2 V9.5, and will be available in late 2007.
- "**DB2 V8 Handbook**"
- "**IBM DB2 Version 9 New Features""

Additionally, see **IBM Redbooks** that contain technical tips on DB2 and related Information Management products.

For business partners, the IBM Virtual Innovation Center (VIC) can be a very useful source of information for IBM products including DB2. To access DB2 Course Materials, on the main VIC page after login, select **All courses by product** under **Shortcuts** at the bottom right. This takes you to a Web page titled "Product Selections." Follow the instructions describing how to access product related course materials.

### Design and implementation tips

#### Utilize the DB2 Migration Tool Kit

When migrating a database from another RDBMS to DB2, utilize the DB2 Migration Tool Kit (MTK) for supported source RDBMSs, which include SQL Server, Oracle, Sybase, and MySQL. Tables, views, indices, RI constraints, user-defined functions (UDFs), and some procedures can be migrated by this tool. The quality of the UDF and procedure migrations depend on the complexity of the source code. An evaluation is required to determine if the migrated UDFs and procedures should be used as a basis for refining the code, or if it is easier to recode from scratch. When using MTK, there are some important data type considerations:

- Always match the DB2 column data type to the application variable data type, or the JDBC set method. This can improve data access efficiencies and potentially eliminate some SQL errors.
- Use the smallest data type that meets the application requirement. For example, use SMALLINT (2 bytes) rather than INTEGER (4 bytes), if appropriate. In some cases, the MTK migrates a source DBMS data type to DECIMAL \((x,0)\). For processing efficiency and possible space savings, consideration should be given to replacing DECIMAL \((x,0)\) data types with INTEGER or BIGINT. However, the resulting datatype from arithmetic computations using INTEGER/BIGINT versus DECIMAL must be considered when making this decision.
Database configuration

1. **Begin with AUTOCONFIGURE:** A good place to begin with database configuration is the Configuration Advisor or the AUTOCONFIGURE command. When determining the configuration, it is important to understand how the database is used and what the performance requirements are for the application that is associated with the database being configured.

2. **Use automatic settings when possible:** There are some configuration parameters that can be set to automatic. When set to automatic, DB2 automatically adjusts these parameters to reflect the current resource requirements for the system.

3. **Alter critical parameters to fit your needs:** When deciding on how to alter the initial database configuration for the needs of a particular system, here are a few important considerations to keep in mind:
   - A default database created by DB2 has a tablespace with a 4K page size. Minimally, create a bufferpool with a 32K page size, a system temporary tablespace with a 32K page size, and a user tablespace with a 32K page size. This ensures that any user tables whose row size exceeds 4K bytes is successfully created into a 32K page size tablespace.
   - Initial configuration values for bufferpools are typically too small for normal production systems. More memory allocated through bufferpools is good in most cases and DB2 default values tend to be minimalist values. With DB2 9, the use of self tuning memory manager (STMM) (mentioned below in the Runtime and manageability tips section) allows DB2 to determine the best buffer pool values.
   - The default locklist size (memory heap used for locks) is 100 4K buffers with DB2 Version 8. Even with low-level testing involving a small number of connections and small volumes of data, this can cause lock escalations from row to table locks, resulting in locking problems. With DB2 Version 8, increase the locklist size to at least 1000, through the command `db2 update db cfg for DBNAME using locklist 1000`. With DB2 9, you can simply use STMM with the locklist parameter value of AUTOMATIC, which should avoid most lock memory problems.

For a detailed discussion on bufferpool and tablespace size considerations, refer to "DB2 Basics: Table Spaces and Buffer Pools" (developerWorks, December 2002).

General guidelines for select, update, and insert efficiency

- **Use APPEND ON when possible:** To increase the efficiency of insert processing, if there is no need to physically cluster data in a table, use APPEND ON in the table definition. Note that reuse of space made available by delete activity or update activity, which changes row size, does not occur until the table is reorganized. For additional insert design tips, refer to "Tips for improving INSERT performance" (developerWorks, March 2004).

- **Review code for use of select *:** In general, the advice is to avoid the use of `select *`. This minimizes the amount of data retrieved to just the columns specified. Additionally with the use of `select *`, changes to the database schema and table definitions may require changes to application code to handle both new columns and removed columns.

- **Keep frequently updated columns grouped together:** When a row is updated, DB2 logs the first changed column through the last changed column, so moving infrequently updated
columns before or after frequently updated columns reduces DB2 logging. This is a minor performance recommendation, so no major rework of an application or database design should be considered to achieve this guideline.

**Utilize SQL stored procedures when it will reduce network overhead**

SQL stored procedures can reduce network overhead by minimizing result set traffic to the client, and stored procedures can also offer the performance benefit of static (pre-prepared) SQL. Other benefits of stored procedures are reduction in client-side processing (although DB server resource usage increases) and DB2 management of the code. Some additional tips on the use of stored procedures:

- Try to keep stored procedures small and simple. Each procedure should do only one thing. Putting multiple types of business logic into a single stored procedure makes it more difficult to tune, modify, and understand.
- When possible, use stored procedures to create reusable "components" of business logic.
- "DB2 SQL PL: Essential Guide for DB2 UDB on Linux, UNIX, Windows, i5/OS, and z/OS, 2nd Edition" contains useful design and coding information for SQL stored procedures

**Maximize concurrency**

Maximizing concurrency is critical for good database performance. Some detailed recommendations are provided:

- There are three registry variables affecting concurrency. These registry variables can improve concurrency, but can also impact the behavior of the application. The recommendation is to enable these registry variables at the beginning of the DB2 development process, so all unit testing through full scale testing occur with these concurrency enhancements in effect. For additional information, refer to the "Evaluate Uncommitted Data" and "Performance Variables" section of the "IBM DB2 Database for Linux, UNIX, and Windows Information Center."
  - DB2_EVALUNCOMMITTED=YES: For V8, beginning with V8.1 FP9 (synonymous with V8.2 FP2), the optimal setting is =YES_DEFERISCANFETCH. With V9, just specify =YES.
  - DB2_SKIPDELETED=ON
  - DB2_SKIPINSERTED=ON
- Choose the isolation level providing the greatest concurrency acceptable to the application. There are several ways to specify the isolation level, such as on the SQL statement (applies to just that statement), using the CURRENT ISOLATION special register (applies to the connection), and on the JDBC connection object (applies to the connection).
- Ensure locklist and maxlocks DB config parms are large enough by monitoring occurrences of lock escalation (through DB2 health monitor, db2diag.log, windows event viewer, or other performance monitor, or utilizing DB2 STMM). Insufficient locklist size results in DB2 attempting to "escalate" many row locks to a single table level lock, causing deadlocks if the escalation fails, and greatly impacting concurrency if the escalation succeeds.
- One exception to the goal of maximizing concurrency is when it is known that only a single connection accesses a table during certain times. Consider using **ALTER TABLE <name> LOCKSIZE TABLE** for the benefit of minimizing locking (and related memory/CPU usage). **USE WITH CAUTION.**
"Lock avoidance in DB2 UDB V8" (developerWorks, September 2005) covers many of these concepts.

**Minimize deadlocks**

1. Minimize deadlocks by always accessing resources in the same order, throughout the application. For example, if one application component will access table A, then table B, then table C, and a second application component needs to access tables A and C, the access sequence of A then C should be followed by the second component.
2. A common cause of deadlocks with DB2 Version 8 is the insufficient size of the locklist database configuration parameter, especially if the default value is used. See the Maximizing concurrency section of this article. If this occurs, increase the size of locklist. The default for DB2 9 is to use the STMM, which adjusts the locklist size to normally prevent attempted lock escalation and deadlocks that might result from this.
3. Ensure dependent tables in a referential integrity (RI) relationship have indexes that exactly match the foreign key.

**Utilize connection pooling**

1. Utilize connection pooling, either application server managed or application managed if not running in an application server environment. Opening and closing connections is an expensive process affecting application or database performance and the use of connection pooling helps eliminate most of that overhead.
2. If a large number of connections are used, investigate the connection concentrator feature of DB2. This conserves memory by permitting fewer DB2 "back end" connections to service the application connections.

**Decide on dynamic vs. static SQL**

Dynamic SQL is in far wider use these days than static SQL. It is typically easier to implement, and with statement reuse the performance is essentially the same. However, there are still some situations where choosing static SQL makes sense, involving security considerations, and maximizing performance for certain OLTP workloads.

For general documentation on deciding when to use static SQL, refer to the DB2 online documentation.

For specific Java™ environment considerations, including the use of SQLJ to create static SQL, see the "Introduction to SQLJ" section of the "IBM DB2 Database for Linux, UNIX, and Windows Information Center."

For specific CLI/ODBC environment considerations, including the use of static profiling to create static SQL, see the "Creating static SQL with CLI/ODBC/JDBC Static Profiling" section of the "IBM DB2 Database for Linux, UNIX, and Windows Information Center."

**Minimize the PREPARE cost of db2 statements**

Use parameter markers to prepare an SQL statement once, and then reuse it many times. For more information, refer to the "Parameter Markers" section of the "IBM DB2 Database for Linux, UNIX, and Windows Information Center."
For very expensive SQL statements, the optimizer may benefit from seeing actual variable values (literals) instead of parameter markers. A simple way to accomplish this is to use literals rather than parameter markers in the code. However, this can cause excessive insert activity into the statement cache, potentially impacting performance and memory usage, since SQL statements differing only by a literal value are considered different statements. For static SQL in a package (like SQL stored procedures), use of parameter markers while still allowing optimization by value can be accomplished through the REOPT ALWAYS bind or precompile options. With DB2 9 and dynamic SQL, REOPT ALWAYS can also be specified through a global- or statement-level optimization profile.

Make JDBC use as efficient as possible

Through the use of some basic principles, it is possible to make JDBC use more efficient. First, ensure the data is bound to parameter markers using the correct setxxx method for the column data-type. This avoids datatype conversion overhead and possible SQL datatype mismatch errors. Second, avoid using scrollable result sets when possible, as the server implements temporary tables to support this, with an impact on performance, particularly with large result sets.

Use Design Advisor to recommend indexes

Once a schema has been developed and is available, use the Design Advisor to recommend indexes using SQL statements as input. This is especially important for large or complex queries, not only the most frequently used queries.

Additionally, rerun the Design Advisor against the dynamic SQL cache populated by a full scale application test. This allows indexes to be recommended based on the actual workload, taking into account the frequency of execution of SQL statements. Ensure runstats has been done prior to the Design Advisor execution.

Another option, which takes a little more effort, but allows viewing the access plan utilizing potential (“virtual”) indexes, is to use the following approach:

1. Issue the SQL statement SET CURRENT EXPLAIN MODE RECOMMEND INDEXES.
2. Execute an SQL statement in the same session, which results in the ADVISE_INDEX table being populated with recommended (“virtual”) indexes.
3. Issue the SQL statement SET CURRENT EXPLAIN MODE EVALUATE INDEXES.
4. Execute the same SQL statement. This now populates the explain tables with the access plan chosen by the optimizer based on real and virtual indexes.
5. Finally, use an explain tool such as db2exfmt to see the access plan utilizing current and recommended indexes.

Improve logging performance

The speed at which DB2 logs are written can be an important DB2 performance factor (depending on the rate of data updates). The following are several tips that help improve the performance of DB2 logging:
• Disable autocommit (which is enabled by default in many application environments, such as JDBC). Every commit causes a synchronous log write to occur. The exception to this recommendation is if the application requires a commit after every SQL statement.
• Don't log large data objects (CLOBs, DBCLOBs, and BLOBs) unless necessary.
• Isolate the DB2 logs onto dedicated disk storage (not shared by anything else). In particular, ensure the logs aren't left in their default location under the instance directory, since by default the database itself is also located here.
• Utilize disk storage with fast write cache for the logs.
• Increase the log buffer size to 256 pages or larger.

Use access plan information to help optimize SQL statements

Learn to use access plan information to understand how to optimize your SQL statements. Access plan describes the access path that the DB2 optimizer has chosen for accessing data for an SQL statement. Visual Explain, which is available through the DB2 Control Center, is the simplest way to view an access plan. The `db2expln` and `db2exfmt` commands provide comprehensive information in a more concise manner and are generally used by those frequently analyzing SQL statements for performance.

`db2expln` is slightly easier to use than `db2exfmt`, as you can simply pass a statement to it to be explained. Also, analyzing all the static SQL in a package is easier done with `db2expln`. `db2exfmt` requires that an SQL statement has been previously explained into these tables, and formats the contents of the information in the EXPLAIN tables. `db2exfmt` is typically used for comprehensive performance analysis, as it provides the most optimizer information.

For more information on using these Explain tools, see the "Visual Explain overview", "db2expln - SQL and XQuery Explain Command", and "db2exfmt - Explain table format command" sections of the "IBM DB2 Database for Linux, UNIX, and Windows Information Center."

Runtime and manageability tips

Dedicated server for database

If possible, have servers dedicated for database use only. This allows database tuning to be performed without concern for unknown additional workload.

Autonomic features of DB2

Consider using the autonomic features of DB2, particularly STMM, available with DB2 9, as well as Automatic Maintenance, available with DB2 Version 8 and DB2 9, (particularly automatic runstats). These features not only reduce the amount of operational effort required to monitor and maintain the database, they also result in a more efficiently tuned database.

Avoid I/O bottlenecks

Whenever possible, follow the best practice of minimally 6-10 disks/CPU and spread everything across as many disks as there are available (indexes, data tablespaces, and temp tablespaces).
Strongly consider using automatic storage, available beginning with DB2 Version 8.1 FP9. Always attempt to place DB2 logs on a separate disk from the tablespaces to avoid I/O contention between these critical database components.

Consider turning off monitoring before deploying

Disabling monitoring before application deployment may save approximately 5 percent in monitoring overhead. For some, the benefit of ongoing monitoring in production is deemed worth the cost. Set monitoring to on when debugging problems or analyzing performance. However, especially if this is a database that will be replicated to dozens or hundreds of devices, turn off monitoring before deploying.

The RUNSTATS command is your friend

Periodically run RUNSTATS when significant change to data volumes or data content occurs. Even if the number of rows in a table remains consistent, changes in data contents can result in different, and hopefully better, access paths using statistics updated by RUNSTATS. Periodically, after RUNSTATS is performed, DB2 packages containing static SQL should be rebound (through the db2rbind command) to ensure the static SQL uses optimal access paths.

The REORGCHK command is another friend

Periodically run REORGCHK, and perform recommended REORGs. Similar to RUNSTATS, you should always run REORGCHK when significant changes to data occur.

Collect snapshot data

Periodically collect snapshot data. Snapshot data can provide a periodic view of bufferpool usage, sort heap usage, locking, and table read/write activity. The views provided of dynamic SQL activity (through the db2 get snapshot for dynamic sql on dbname command) and of overall database activity (through the db2 get snapshot for database on dbname command) (replacing dbname with the real database name) are especially useful. By recording snapshots and analyzing them on a periodic basis, you can spot both unusual activity and trends that occur. This periodic use of snapshots can be especially useful in detecting trends. For particularly easy longer-term monitoring, consider using the SQL table functions to access snapshot data. Storage and comparison of many snapshots is much easier when the data is back inside DB2.

Learn to use the db2pd tool

This is a useful tool for both problem determination and for general analysis of what is happening in DB2. It can also be used to monitor DB2 log usage as well as DB2 bufferpool usage.

Learn to use the DB2 catalog

The DB2 catalog is a valuable source of information. The recommended user access to the catalog is through a set of views with the schema of SYSCAT. In DB2 9, useful information about the status and usage of tables can be found by using a new view called
SYSIBMADM.ADMINTABINFO. For more information on the DB2 catalog refer to the "Roadmap to the Catalog Views" section of the "IBM DB2 Database for Linux, UNIX, and Windows Information Center."

Apply DB2 fix packs at regular intervals

Keeping the DB2 product fixes up-to-date is a critical step in preventing problems. Schedule the application of DB2 fixpacks at regular intervals, even if there are no errors in the system. Like any software product, DB2 is not perfect and IBM constantly tries to improve the reliability of software with releases of fix packs nearly every quarter. Applying fix packs allows you to be proactive in preventing potential errors, and it also helps minimize the potential downtime of DB2 related application systems.

Conclusion

This article has provided you with basic design, implementation, runtime, and manageability guidelines to assist with implementing your application in DB2. Although it has not provided all the details necessary to implement these guidelines, hopefully this article has provided you with awareness of important considerations. Utilize the additional resources mentioned (found in the Related topics section), to obtain further technical guidance on the items presented in this article.

Acknowledgements

We would like to thank Bill Wilkins and Steve Rees for their helpful review of this article.
Related topics

- Information Management developerWorks Technical Library: Find a wide range of content, including technical articles, product documentation, and tutorials.
- "Understand DB2: Learning Visually with Examples": Learn DB2 Version 8.2 visually, step by step, with dozens of examples and case studies drawn from the authors' unsurpassed experience as DB2 consultants at IBM. The 2nd edition of this book has been written for DB2 9.5, and will be available in late 2007.
- "DB2 V8 Handbook": Read a start-to-finish guide to DB2 Universal Database V8 administration and development for UNIX, Linux, and Windows.
- IBM Redbooks: Find technical tips on DB2 and related Information Management products.
- "IBM DB2 Version 9 New Features": Learn about DB2 9.
- "DB2 Basics: Table Spaces and Buffer Pools" (developerWorks, December 2002): Understand the importance of table spaces and bufferpools, and explains how properly designing and tuning them can enhance database performance.
- "Tips for improving INSERT performance" (developerWorks, March 2004): Find out exactly what happens when an insert occurs, look at alternatives, and examine issues that affect insert performance, such as locking, index maintenance, and constraint management.
- "Lock avoidance in DB2 UDB V8" (developerWorks, September 2005): Gain an understanding of how row locking works in DB2 UDB, and learn how you may be able to improve concurrency of scans by deferring row locking until after it is determined that a row qualifies for a query.
- Download IBM product evaluation versions and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

© Copyright IBM Corporation 2007
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)