IBM Software Group

Introduction to the WebSphere Commerce Database for DB2 Administrators

Andres Voldman (voldman@ca.ibm.com)
WebSphere Commerce Support
5 August 2010

WebSphere® Support Technical Exchange

Agenda

- DB2 Versions supported by WebSphere Commerce
- WebSphere Commerce Schema
- WebSphere Commerce Database Creation
- Connection Management
- Locking
- Staging Server
- Database Maintenance
- Performance

Supported DB2 Versions

- WebSphere Commerce Version 7.0
  - DB2 Universal Database Express/Enterprise Edition Version 9.5.4
  - DB2 Universal Database Express/Enterprise Edition Version 9.7.1 (Requires 7.0.0.1)
  - DB2 Universal Database Version 9.8 (PureScale) - limited support
- WebSphere Commerce Version 6.0
  - DB2 V8.2 Fix Pack 3
  - DB2 V9.1 Fix Pack 3 Server with DB2 8.2 Client (Requires 6.0.0.6)
  - DB2 v9.5 Fix Pack 1 Server with DB2 8.2 Client (Requires 6.0.0.6)
- WebSphere Commerce Version 5.6.1
  - DB2 V8.2 Fix Pack 1

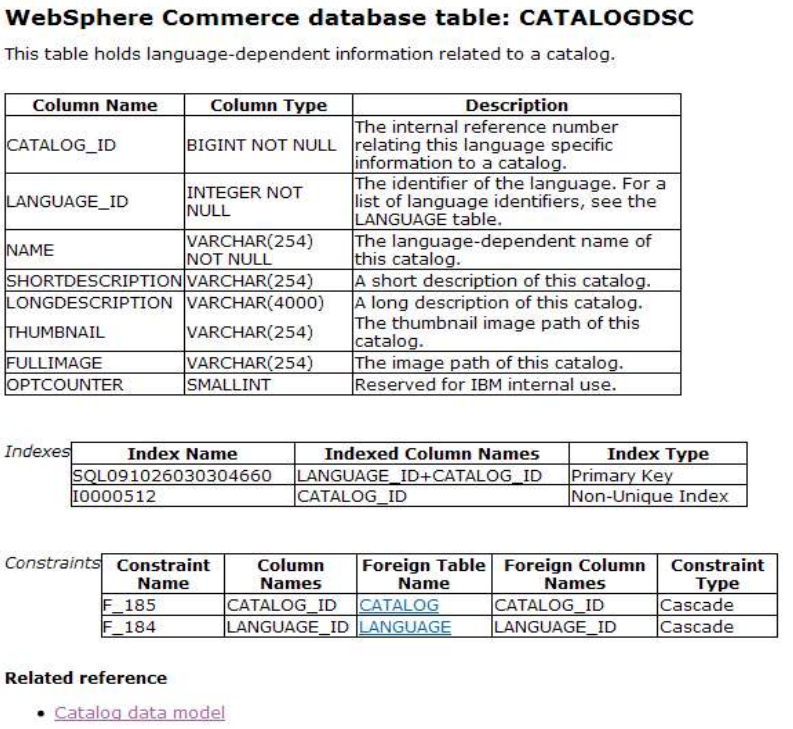
Supported DB2 Versions

- All DB2 Fix Packs that are higher than the minimum requirements are supported
  - WebSphere Commerce Version 7 software requirements
  - List of supported software for WebSphere Commerce Version 6.0
  - List of supported software for WebSphere Commerce Version 5.6.1
- DB2 Client for remote database configurations
  - The Admin client version is required
  - The client instance needs to match the bits of the Java™ runtime
  - WebSphere Commerce has 64 bit support starting with version 7.0.0.1
  - WebSphere Commerce 5.6.1 and 6.0 use the Type 2 Legacy Driver
  - WebSphere Commerce 7.0 uses the Universal Driver with Type 4
    - But invokes functions that still require the client

Schema

WebSphere Commerce Schema

- The WebSphere Commerce Schema contains all the data relevant to the running of an e-commerce site:
  - Customer, Catalog, Pricing, Inventory, Orders etc
- The schema is well documented in the WebSphere Commerce Information Center
  - WebSphere Commerce Tables
  - Data Models
- For each table, the documentation includes
  - Short usage description
  - Column names, types and usage
  - Indexes
  - Foreign keys



WebSphere Commerce database table: CATALOGDSC

This table holds language-dependent information related to a catalog.

Column Name	Column Type	Description
CATALOG_ID	BIGINT NOT NULL	The internal reference number relating this language specific information to a catalog.
LANGUAGE_ID	INTEGER NOT NULL	The identifier of the language. For a list of language identifiers, see the LANGUAGE table.
NAME	VARCHAR(254) NOT NULL	The language-dependent name of this catalog.
SHORTDESCRIPTION	VARCHAR(254)	A short description of this catalog.
LONGDESCRIPTION	VARCHAR(4000)	A long description of this catalog.
THUMBNAIL	VARCHAR(254)	The thumbnail image path of this catalog.
FULLIMAGE	VARCHAR(254)	The image path of this catalog.
OPTCOUNTER	SMALLINT	Reserved for IBM internal use.

Indexes

Index Name	Indexed Column Names	Index Type
SQL091026030304660	LANGUAGE_ID+CATALOG_ID	Primary Key
I0000512	CATALOG_ID	Non-Unique Index

Constraints

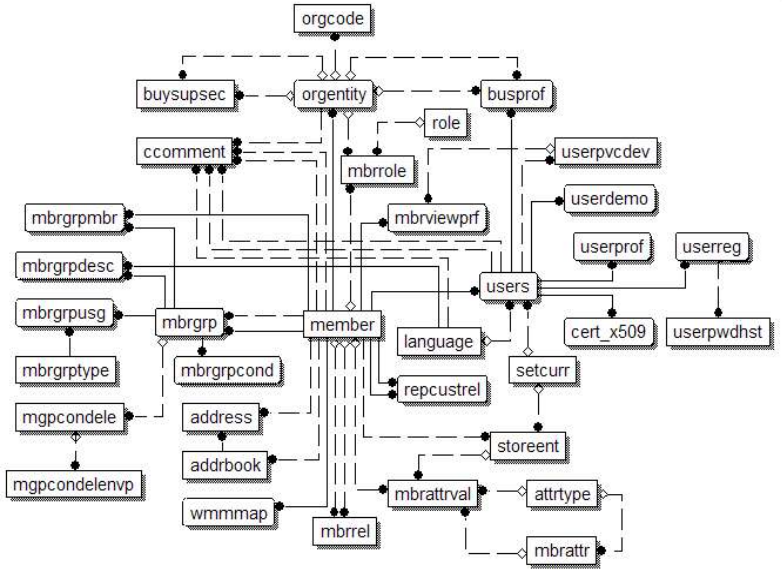
Constraint Name	Column Names	Foreign Table Name	Foreign Column Names	Constraint Type
F_185	CATALOG_ID	CATALOG	CATALOG_ID	Cascade
F_184	LANGUAGE_ID	LANGUAGE	LANGUAGE_ID	Cascade

Related reference

- Catalog data model

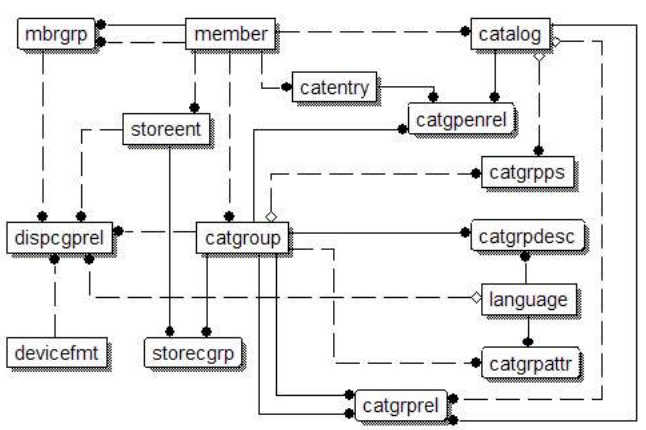
Key Data Models

- The Member Data Model maintains user and organization information
- As the number of guest and registered customers increases, these tables will become larger
  - MEMBER, ORGENTITY, USERS, USERREG, MBRREL, MBRROLE, etc
- Most assets are “owned” by a particular organization or user; many tables have Foreign Keys to the MEMBER table

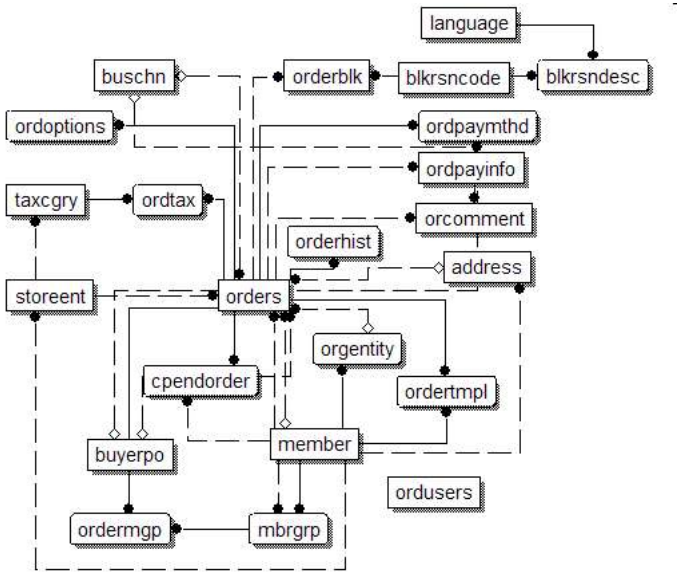


Key Data Models

- The Catalog Data Model maintains information for categories, products and attributes and how they relate
  - CATENTRY, CATENTDESC, CATGROUP, CATGRPDESC, etc



- The Order Management Data Model contains information for orders, calculations, ATP inventory, currency, offers etc
  - ORDERS, ORDERITEMS, OFFER, OFFERPRICE, etc



Customizing the Schema

- Although certain changes to the out-of-the-box schema are supported, others are restricted to prevent issues in the code or during migration
- The following page documents examples of supported and unsupported schema changes
  - WebSphere Commerce database schema
- Example of supported changes:
  - Changing the tablespaces
  - Adding Indexes
- Examples of unsupported changes:
  - Adding / dropping or renaming columns on out-of-the-box tables
  - Changing primary keys

WebSphere Commerce Database Creation

WebSphere Commerce Database Creation

- WebSphere Commerce Instance creation is the process to create a WebSphere Commerce server with its associated database
- During the creation of the WebSphere Commerce instance the following options are supported
  - Create a new remote or local database
  - Use an existing local or remote database
    - Create an new schema
    - Use an existing schema
      - The existing schema should have tables but no referential constraints or data
- It is possible to use a populated existing database but it requires some “hacking” of the Instance creation Ant scripts

WebSphere Commerce Database Creation

- During instance creation, the following .sql files are executed: (among others)
  - *CommerceServer*\schema\db2\wcs.schema.sql
  - *CommerceServer*\schema\db2\wcs.key.sql
  - *CommerceServer*\schema\db2\wcs.index.sql
- You could customize these files to, for example, specify a different tablespace
- Keep in mind that these are not the only files that contain DDL
  - Applying a fix pack or a feature pack could also modify the schema and create new tables

Database and Storage Defaults

- The createdb.db2.(sh/bat) file in the bin directory contains the default options that will be used when creating a new database
  - You should review it to get familiar with the configuration
  - This file is executed during instance creation if you use the new db option
- Default Bufferpools

BUFF8K	50	8K	AUTOMATIC
BUFF16K	50	16K	AUTOMATIC
BUFF32K	50	32K	AUTOMATIC
IBMDEFAULTBP	2000	4K	AUTOMATIC

- Default Tablespaces

REGULAR	TAB8K	8K
REGULAR	TAB16K	16K
TEMPORARY	TEMPSYS8K	8K
TEMPORARY	TEMPSYS16K	16K
TEMPORARY	TEMPSYS32K	32K

Stored Procedures

- Most WebSphere Commerce logic is in Java code
- Only a few Stored Procedures are created out-of-the-box which are used for the ATP (available-to-promise) inventory model
    - AVAILRECEIPTS, EXPECTEDINV, GETITEMS etc
- It is important to rebind the store procedures from time to time to ensure they take advantage of the latest stats

Connection Management

Connection Management

- Each WebSphere Commerce server will use a pool of connections
  - When a request is received, a connection is taken from the pool. When the request is complete, the connection is returned to the pool
  - As a pool is used, you will typically see a large number of established connections but most of them will be in “UOW waiting” (Unit of Work waiting = idle) state
  - The WebSphere Commerce server contains multiple configuration options to control the pool and when connections should be discarded
  - The maximum number of connections per server * number of servers will determine the maximum number of connections to the database
    - You should find out what this number is and tune for it

Connection Management

- What makes the number of connections increase?
  - During warm up, you will see more connections until the cache is primed
  - A server slowdown will increase the number of connections
    - For example, on a typical day the server does not need more than 5 connections at the time.If the server slows down, when new requests come in, instead of finding free connections, as requests are taking longer than usual, a new connection is made available. The number of connections increases up to the maximum allowed
  - If a connection is not used for some time, the server will close it (inactivity time-out)

Transaction Management

- WebSphere Commerce automatically starts a new DB transaction for every request that is received
  - For example, a product page is requested from the browser or an item added to the shopping cart
- At the end of the request the transaction will be committed or rolled back and returned to the pool
- If the request is slow for any reason, database locks will be held longer
  - A cascade effect is created
  - It could lead to lock time-outs or deadlocks

Locking

Isolation Levels

- WebSphere Commerce 5.6.1 uses RS (Read Stability)
  - All write locks are kept until the end of the transaction
  - Shared locks for selected rows are kept until the end of the transaction
- WebSphere Commerce 6.0 and 7.0 use CS (Cursor Stability)
  - Only write locks are kept until the end of the transaction
  - In order to lower the isolation, WC 6.0 and 7.0 implement optimistic locking
- The Isolation Level for out-of-the-box code cannot be changed

Optimistic Locking (WC 6.0 and 7.0)

- Optimistic Locking is used to lower the isolation level
  - All the tables have a special column called “optcounter”
  - When a row is selected, the optcounter value is saved
  - If the row is updated, the original optcounter value is used in the WHERE clause and a new optcounter is set

    UPDATE TABLE myTable SET col1 = ‘a’, col2 = ‘b’, optcounter = 101
    WHERE pk = 10001 AND optcounter = 100
  - Because the optcounter is used in the WHERE clause, if the row is changed (and the optcounter increased) by another connection, the UPDATE will return ‘no rows updated’ and the application will receive an OptimisticUpdateFailureException

Optimistic Locking Triggers

- In order to ensure that the optcounter is increased when a row is updated, a trigger is defined for every table

```
CREATE TRIGGER perf_mem
    NO CASCADE BEFORE UPDATE ON member
    REFERENCING NEW AS NEWOPT OLD AS OLDOPT
    FOR EACH ROW MODE DB2SQL
     when ((NEWOPT.optcounter is null) or (OLDOPT.optcounter =
NEWOPT.optcounter))
      begin atomic
            set NEWOPT.optcounter = case
             when OLDOPT.optcounter < 32767
             then
                 OLDOPT.optcounter + 1
             else
                 1
             end;
      end
```

Locking

- When dealing with locking, it is important to understand the nature of the connections involved
- The most common scenarios are
  - Locking between store front requests
  - Locking between store front requests and a batch/back-end process
- To distinguish store front requests versus back-end/batch requests you need to be familiar with the schema
  - The query itself will give you a clue. For example if it is for a particular ID or a group of IDs
  - The snapshot will show the JVM (hostname and port) where the request is coming from
  - If the query times-out or deadlocks, the WebSphere Commerce system log will show the stack trace of the calling method
  - Time of day and interval in which the locking occurs is also a clue

Locking between store requests

- Multi-click scenario
  - As a new transaction is created for each Web request it is possible for a single user to have more than one connection running at the same time
    - These connections will typically deadlock, time-out or result in optimistic locking exceptions
  - There are server and client techniques to prevent multiple executions but they are not fail proof
  - Most Commerce installations are configured with session affinity, so you should see both requests coming from the same JVM (hostname and port in the snapshot)

Locking between store requests

- Different users locking each other
  - Look at the tables involved and the nature of the queries
  - Does it make sense for those tables to lock? Is it normal for lock waits to happen between those requests?
  - For example, lock waits on two transactions trying to update inventory for the same product is expected
  - On the other hand, two different users locking on the orders table when querying their own orders is a problem
    - Statement might be table scanning or not using the correct indexes

Locking between back-end/batch and store requests

- Back-end and batch updates can be a source of contention because they can operate on multiple rows and on data that is being used by the store front
  - Staging propagation
  - Data loads
  - Data cleanups
  - Cron / Scheduled Jobs
- This situation might be difficult to resolve as the database is working as designed
  - Batch processes should run during times of low activity (when possible)
  - Smaller units of work for the batch process help concurrency
  - Implement an application specific lock time-out (longer) or re-tries
  - Caching in the Commerce server can help maintain the site running

DB2_WORKLOAD=WC

- DB2_WORKLOAD=WC is available since DB2 9.5 Fix Pack 4
- It is set by default with WebSphere Commerce 7
- When set, it will expand and configure the following variables, several of which help reduce locking

```
C:\>db2set -gd DB2_WORKLOAD=WC
DB2_REDUCED_OPTIMIZATION=INDEX,UNIQUEINDEX,JOIN,NO_SORT_MGJOIN,JULIE
DB2_MINIMIZE_LISTPREFETCH=YES
DB2_INLIST_TO_NLJN=YES
DB2_ANTIJOIN=EXTEND
DB2_EVALUNCOMMITTED=YES
DB2_SKIPINSERTED=YES
DB2_OPTPROFILE=YES
DB2_OPT_MAX_TEMP_SIZE=10240
DB2_SKIPDELETED=YES
```

- If your version of DB2 is earlier than 9.5.4 you can set the variables individually
  - DB2 Registry variables which help reduce locking in a WebSphere Commerce database

Improved Schema for DBClean in WC 7

- To reduce locking that can occur during delete operations, DB2 requires that all the foreign keys be supported by indexes
  - Referential Integrity and Julie's Scenario
- In WebSphere Commerce Version 7, the out-of-the-box schema has been extended to include additional indexes on foreign keys to minimize locking on delete operations
- DB2 has also made enhancements for locking which are available starting with DB2 9.5 Fix Pack 4
  - IZ16959: OPTIMIZER NOT CHOOSING CORRECT INDEX DURING REFERENTIAL INTEGRITY CHECKING
  - The DB2 APAR IZ16959 to support delete operations is enabled with the JULIE setting which is part of DB2_WORKLOAD=WC

```
DB2_REDUCED_OPTIMIZATION=JULIE
```

Staging Server

Staging Server

- WebSphere Commerce can be used as a Staging Server
  - Using triggers, changes are captured into the STAGLOG table
  - When the stagingprop utility runs, it reads all the changes from the STAGLOG table and executes them in production
    - The STGMERTAB, STGMRSTTAB and STGSITETAB tables list the tables to be propagated. For example, the ORDERS table is never propagated as staging does not maintain user data
  - If the databases are out of sync (e.g. data is directly updated in production) the stagingprop process can fail
- The stagingcopy utility can be used to synchronize the staging server with data from the production server
  - It should only be used during initial setup and special circumstances
  - It does not copy user and order data

Staging Triggers

- Tables in the staging environment have a trigger to populate the STAGLOG table similar to the following:

```
Create  trigger STAG0031
            AFTER INSERT ON qtyunit REFERENCING NEW AS N
            FOR EACH ROW MODE DB2SQL
            BEGIN ATOMIC
    INSERT INTO staglog
    (  stgrfnbr,  stgstmp,  stgtable,  stgop, stgmenbr,  stgkey3name,  stgokey3, stgnkey3      )
    VALUES (
       NEXTVAL FOR STAGESEQ
     ,  CURRENT TIMESTAMP
     ,  'qtyunit' ,'I',  0,  'qtyunit_id',  N.qtyunit_id,  N.qtyunit_id
    );
            END#
```

- The use of a sequence is new to V7. Before V7 the KEYS table was used instead
- Triggers will need to be created for custom tables as described in the Creating triggers for custom tables document

Staging Server Considerations

- The staging triggers capture data changes as-is
  - If the databases are not in sync, the propagation process will fail
    - Duplicate key exceptions, missing foreign key relations,updates on data that does not exist in production, etc
- You should plan carefully how the databases will be updated
  - When Staging is used, most updates should be done in Staging and not production
  - Some updates can be done directly to production (such as price update feeds) but you need to ensure the same data is also not manipulated in Staging

Workspaces

- Workspaces was introduced with WebSphere Commerce 6.0
- It allows users of the staging server to work in their own schema
  - Once the user is satisfied and the changes are approved, the changes are moved to the base schema from where can be propagated to production
- When an instance is created with the workspaces option you will find multiple schemas are created
  - Each workspace will be associated to one read and one write schema
- Special steps are required to include custom tables in the workspace
  - Enabling workspaces support for a custom table in an existing WebSphere Commerce BOD service module
  - Techniques for improving the performance of SQL queries under workspaces in the Data Service Layer

Workspaces

- Write schemas; named WCW* - e.g. *WCW101*
  - It contains actual tables that hold the information that is created and updated under the workspace
- Read schemas; named WCR* - e.g. *WCR101*
  - Instead of tables, the read schemas contain views that return the contents of the write schema plus the base schema
  - As these are not real tables, when using the Store Preview functionality with workspaces you might find that the queries perform slower than expected

```
CREATE VIEW WCR101.LISTPRICE (CATENTRY_ID, CURRENCY, LISTPRICE, OID, OPTCOUNTER)
AS SELECT CATENTRY_ID, CURRENCY, LISTPRICE, OID, OPTCOUNTER
    FROM DB2INST1.LISTPRICE
   WHERE NOT EXISTS (
  SELECT 'match'
    FROM WCW101.LISTPRICE
   WHERE DB2INST1.LISTPRICE.CURRENCY    = WCW101.LISTPRICE.CURRENCY
     AND DB2INST1.LISTPRICE.CATENTRY_ID = WCW101.LISTPRICE.CATENTRY_ID )
   UNION ALL
  SELECT CATENTRY_ID, CURRENCY, LISTPRICE, OID, OPTCOUNTER
    FROM WCW101.LISTPRICE
   WHERE CONTENT_STATUS <> 'D'
```

Database Maintenance

Database Maintenance

- Standard database maintenance tasks are required for the Commerce Database
  - RUNSTATS
  - REORG
  - REBIND
- These tasks are usually done weekly during low traffic periods
  - They could also be done more frequently depending on batch operations

Database Maintenance

- The database will grow fast and without a strategy in place, lack of data maintenance will introduce problems in a matter of weeks
  - Performance degradation
  - Increased backup and restore times
  - Exceeding the available storage
- Besides user information (users, orders, etc) that grows every day, the WC site will store session data that needs to be cleaned daily or weekly to avoid problems
- The following Webcast covers the topic in detail
  - Webcast replay: Data Maintenance in a WebSphere Commerce Database

Performance

Performance

- Commerce works mainly as an OLTP Database
- Most queries use the primary key or return a small result set
- Queries are supposed to be FAST as they might be executed a very large number of times

Calculated Columns for Search Performance in V7

- In WebSphere Commerce v7, new calculated columns have been added to the out-of-the-box schema
  - Calculated (Generated) columns are created to store the UPPER() version of an alphanumeric field
  - The column is indexed
  - DB2 will automatically detect the calculated column and it will use it when the SQL statement uses the UPPER() function on the column
  - This will improve performance with case sensitive search

TABNAME	COLNAME	RULE
ACCOUNT	UP_NAME	AS (UPPER(NAME))
ATCHAST	UP_ATCHASTPATH	AS (UPPER(ATCHASTPATH))
ATCHTGT	UP_IDENTIFIER	AS (UPPER(IDENTIFIER))
ATCHTGTDSC	UP_NAME	AS (UPPER(NAME))
CATENTDESC	UP_NAME	AS (UPPER(NAME))
CATENTRY	UP_MFNAME	AS (UPPER(MFNAME))
CATENTRY	UP_MFPARTNUMBER	AS (UPPER(MFPARTNUMBER))
CATENTRY	UP_PARTNUMBER	AS (UPPER(PARTNUMBER))
CATGROUP	UP_IDENTIFIER	AS (UPPER(IDENTIFIER))
CATGRPDESC	UP_NAME	AS (UPPER(NAME))
CATGRPDESC	UP_SHORTDESC	AS (UPPER(SHORTDESCRIPTION))
COLLATERAL	UP_NAME	AS (UPPER(NAME))

Application Query Caching

- You can help the WebSphere Commerce Administrators find opportunities for caching
- Being familiar with the schema will help you find good candidates such as configuration tables
- Use a snapshot and rank the top X (e.g. 20) queries by number of executions in a 10 minute period
  - Review the statements. Are there statements that could potentially be cached?
    - Tables that just contain a few hundred rows but are still queried thousands of times
    - For example, do you think queries to the ORDERS table could be cached?
    - What if you are seeing thousands of queries to the STORE table but your site only contains a few stores?

References

- WebSphere Commerce Version 7 Information Center
- IBM® DB2 Database for Linux®, UNIX®, and Windows® Information Center
- Webcast replay: Data Maintenance in a WebSphere Commerce Database
- DB2 Registry variables which help reduce locking in a WebSphere Commerce database

Additional WebSphere Product Resources

- Learn about upcoming WebSphere Support Technical Exchange webcasts, and access previously recorded presentations at:
  http://www.ibm.com/software/websphere/support/supp_tech.html
- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at:
  http://www.ibm.com/developerworks/websphere/community/
- Join the Global WebSphere Community:
  http://www.websphereusergroup.org
- Access key product show-me demos and tutorials by visiting IBM Education Assistant:
  http://www.ibm.com/software/info/education/assistant
- View a webcast replay with step-by-step instructions for using the Service Request (SR) tool for submitting problems electronically:
  http://www.ibm.com/software/websphere/support/d2w.html
- Sign up to receive weekly technical My Notifications emails:
  http://www.ibm.com/software/support/einfo.html

We Want to Hear From You!

Tell us about what you want to learn

Suggestions for future topics
Improvements and comments about our webcasts
We want to hear everything you have to say!

Please send your suggestions and comments to:
wsehelp@us.ibm.com

Questions and Answers