Getting Started with SAP Sybase IQ Column Store Analytics Server

Lesson 4: Optimize a Query using the SAP Sybase IQ Query Plan



Table of Contents

1.	Introduction		
2.	SAP Sybase IQ Query Processing		
3.			4
4.	l. What a Query Plan Looks Like		
5.	What	a Query Plan Will Tell You	7
5	5.1 Oı	uery Tree Nodes and Node Detail	7
		Root Node Details	
		Leaf Node Details	
	5.1.3	Join Node Details	10
	5.1.4	Group Node Details	10
6.	What	Steps Can You Take to Optimize a Query	12
7.	Actua	l Example of Improving a Query	13
7	7.1 Tł	ne query	13
		urn on the Query Plan and the Index Advisor	
7		dex Advice	
R	Summ	nary	16

1. Introduction

In this chapter, you will learn how SAP Sybase IQ executes queries, and how to interpret the SAP Sybase IQ query plan. The focus will be on query optimization on a single IQ server, not distributed query processing in a Multiplex. That is addressed in another lesson, "Scaling Out with Multiplex and Distributed Query Processing". In this lesson, you will execute a SQL query, and perform some analysis to optimize the query.

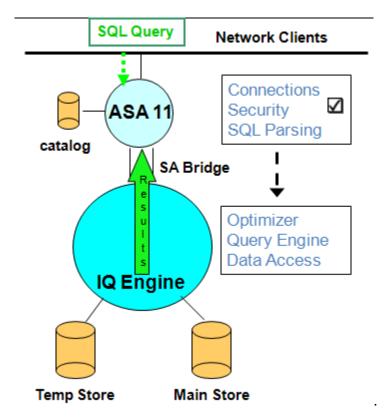


Ready to query some data.

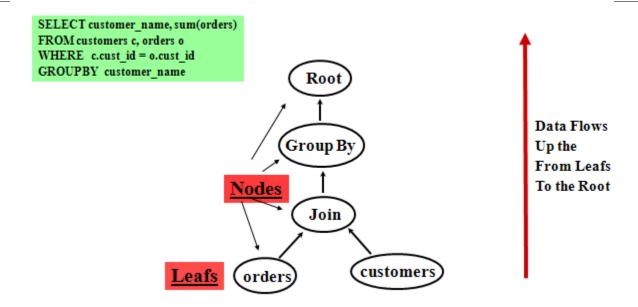
2. SAP Sybase IQ Query Processing

SAP Sybase IQ includes as part of its architecture a small footprint relational DBMS called SQL Anywhere, or ASA. ASA is a product that can exist with or without SAP Sybase IQ. SAP Sybase IQ uses ASA as a front end for various functions, and to manage the catalog, or metadata, for SAP Sybase IQ. The way that IQ processes a query is as follows:

- ASA handles connections, security and SQL parsing
- The query is handed off to the IQ engine for optimization and execution
- The result set is sent back through ASA to the client



When SAP Sybase IQ optimizes a query, it builds an upside down "tree" of objects (joins, group by clauses, subqueries, etc.) Tables are "leaves" at the bottom of the tree, and rows of data flow up the tree from the leaves to a "root" node at the top, where the data is passed out of SAP Sybase IQ to the user. This model with a tree of processing nodes is called a "data flow" model:



In SAP Sybase IQ, this tree of data flow objects starts executing at the root node at the top of the tree. This node starts by requesting a first row from the next object below. The node below then "wakes up", and unless the node contains a base table, it begins asking for rows from the next node below it. This continues down the tree until the execution reaches the database tables, which read the data in from disk.

A database table in SAP Sybase IQ is accessed through a leaf node. A leaf represents two functions in a SAP Sybase IQ query. It first contains those local table predicates – that is, the parts of the WHERE clause that access only one table – which can be processed "vertically", or column by column, within the SAP Sybase IQ indexes. Once the set of rows relevant to the remainder of the query has been identified (because they satisfied all the local predicates), then the second function is to project that set of rows from the table up to the next higher node operation in the tree.

3. SAP Sybase IQ Query Plans

Proper index selection is essential for optimal query performance in a SAP Sybase IQ database. Unlike traditional relational database management systems, nearly every column in a SAP Sybase IQ database will be indexed, and some will have multiple indexes. Within a single leaf node in a query, SAP Sybase IQ may actually use many indexes on different columns in the same query. Having multiple indexes on the same column gives the optimizer the opportunity to select the best index for an operation, given the currently available resources.

For understanding how the SAP Sybase IQ engine processes a query, the most useful tool is the *query plan*. Query plans provide important details about how a query was executed (or will be executed). Use a query plan when:

- You suspect that a query is running poorly
 - Query plans show you how the server processed the query
 - You may be able to identify a problem in the plan
- You are not sure whether you have the correct indexes for a query
 - Query plans tell you what indexes are being used
 - The Index Advisor can provide additional guidance

You can access the query plan through the following mechanisms:

- In the IQ message file (IQ's user log file), as text embedded in IQ message log entries
- On the server machine as an HTML file
- From the client as a graphical plan:
 - o Graphical image displayed in the dbisql database access tool
 - o As HTML
 - o As XML

Query plans are not generated by default. You need to turn on various database options to cause them to be created. You will set these options later in this chapter.

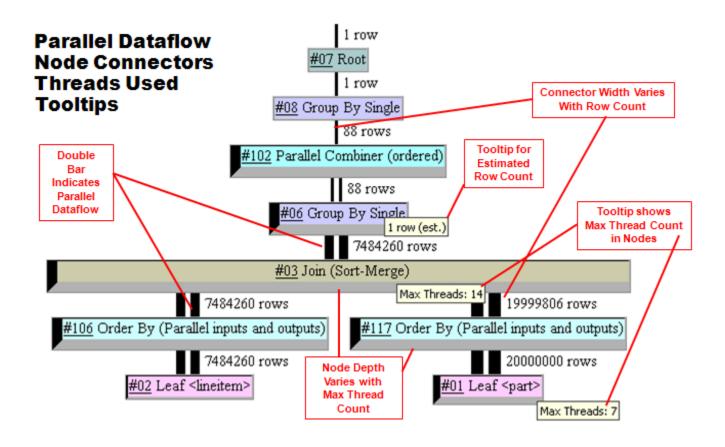
4. What a Query Plan Looks Like

A query plan is represented as a query tree, with an associated timing diagram, and detail about each node in the tree. The query tree consists of nodes representing query execution steps and connections between nodes where data flows.

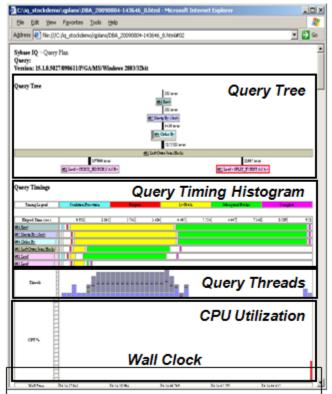
In the query tree, the node type indicates the type of operation performed by that query plan node – for example a join, group by, subquery, etc. Sometimes it also indicates the type of algorithm being used to perform that operation.

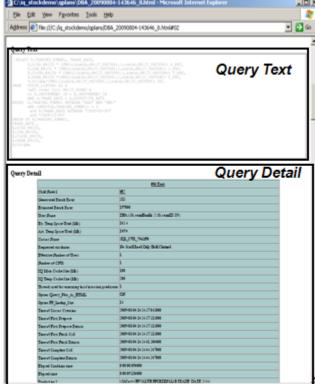
In the query tree, nodes are color coded and numbered. Connections between nodes show row counts flowing between nodes 'up' the query tree.

The top node in the tree is the *root node*. *Leaf nodes* represent tables, typically at the bottom or edges of the tree. When a node is being executed in parallel, double lines will connect that node with the one above it. The node depth (the node rectangle is represented in 3D) shows the relative number of threads used. The node connector width is relative to the number of rows flowing between nodes. In the graphical view of a query tree, hovering the mouse over parts of the plan reveals notes. Nodes containing advice about missing indexes are highlighted.



When you have all query plan options turned on, you see not only the query tree, but a timing diagram, query thread and CPU utilization, and query detail. If you click on a node in the query tree, the detail for that node will display. Here is a depiction of a complete query plan:





5. What a Query Plan Will Tell You

We will concentrate on the query tree and node details for this discussion.

5.1 Query Tree Nodes and Node Detail

There are a large set of node types that you will see in a SAP Sybase IQ query tree. The ones you should pay attention to are the following, however:

- Root node
- Leaf node
- Join node
- Group node

These nodes will give you the most information about how the query is being processed, and guide you as to how you can improve it. Note that the node details shown below are based on a query plan generated after the query was executed.

5.1.1 Root Node Details

The root node is the top node of the tree:

#14 Root		
Child Node 1	#1 <u>5</u>	
Generated Result Rows	9	
Estimated Result Rows	1823229	
User Name	DBA (SA connHandle: 8 SA connID: 3)	
Est. Temp Space Used (Mb)	574.4	
Act. Temp Space Used (Mb)	338.8	
Cursor Name	SQL_CUR_1abcd168	
Requested attributes	Scroll Read Only Hold Chained	
Effective Number of Users	1	
Number of CPUs	2	
IQ Main Cache Size (Mb)	400	
IQ Temp Cache Size (Mb)	600	
Threads used for executing local invariant predicates	1	
Option Query_Plan_As_HTML	ON	
Option Max_Hash_Rows	5000000	
Option FP_Lookup_Size	18	
Option Hash_Pinnable_Cache_Percent	40	
Time of Cursor Creation	2010-05-27 11:10:59.677000	
Time of First Prepare	2010-05-27 11:10:59.880000	
Time of First Prepare Return	2010-05-27 11:10:59.880000	
Time of First Fetch Call	2010-05-27 11:10:59.880000	
Time of First Fetch Return	2010-05-27 11:11:07.148000	
Time of Complete Call	2010-05-27 11:11:18.474000	
Time of Complete Return	2010-05-27 11:11:18.474000	
Elapsed Condition time	0:00:00.125000	
Working time	0:00:07.489000	
Elapsed time	0:00:16.797000	

The important items to be checked in the root node are the following:

- 1. Generated Result Rows: these are the actual number of result rows returned after executing the node
- 2. Estimated Result Rows: the estimated number of result rows which will be created after executing the node
- 3. (The actual and estimated numbers of items 1 and 2 above should be close to the same. If not, then it indicates that the optimizer does not have enough metadata defined primary and foreign keys, and sufficient indexes)
- 4. Estimated and Actual Temp Space Used (Mb): "temp space" refers to the amount of temp cache used
- 5. Effective Number of Users: the count of concurrent queries occurring at the time this query is executing
- 6. Number of CPUs: this is the number of CPUs that IQ believes are available
- 7. Working time: amount of CPU time expended processing query
- 8. Elapsed time: amount of clock time expended processing query

5.1.2 Leaf Node Details

A leaf node represents access to a table in the IQ store, and provides row counts, and information to process the query condition (WHERE clause). For each predicate (condition in the WHERE clause that filters one table), the query optimizer determines:

- Selectivity: the portion of the table that satisfies a given predicate. The selectivity is determined
 by the metadata (count of rows that have a particular value) provided by the HG, LF and FP
 indexes
- Cost: best index to resolve the predicate, based on resources to use the index
- Usefulness: the best order to execute the predicates in this table.
 - o The usefulness value range is 0.0 to 10.0, with 10.0 being the most useful
 - The predicate with the highest usefulness value is executed first
 - o The remaining predicates are executed in descending order

When NO metadata is available to determine the exact selectivity of a predicate, the optimizer estimates the selectivity based on the predicate operator:

Predicate Type	Selectivity
Equality (=)	0.2000000
Range (>, >=, <, <=)	0.4000000
Between	0.4000000
Like (%)	0.2000000
Inter-column equality $(t.x = t.y)$	0.3000000
Inter-column range (e.g., t.x <= t.y)	0.5000000

If you see a selectivity with one of the values shown in this table, then you have a good clue that you are missing metadata – a proper index on the column to help the optimizer know how many rows in the table have a particular value.

Here is an example of a leaf node:

#02 Leaf		
Table Name	TPCD.ORDERS	
Parent Node	#38	
Table Row Count	1500000	
Generated Result Rows	1139288	
Estimated Result Rows	1139288	
Generated Post Invariant Predicate Rows	1139288	
Estimated Post Invariant Predicate Rows	1139288	
Invariant Predicate Thread Allowance	7	
Parallel Source Work Units	2	
Initial Source Work Unit size	750000	
Time of Condition Start	2010-05-27 11:10:59.708000	
Time of Condition Done	2010-05-27 11:10:59.833000	
Time of First Prepare	2010-05-27 11:10:59.880000	
Time of First Prepare Return	2010-05-27 11:10:59.880000	
Time of First Fetch Call	2010-05-27 11:10:59.880000	
Time of First Fetch Return	2010-05-27 11:10:59.880000	
Time of Complete Call	2010-05-27 11:11:16.474000	
Time of Complete Return	2010-05-27 11:11:18.474000	
Elapsed Condition time	0:00:00.125000	
Elapsed time	0:00:16.797000	
Declared Primary Key Column 1	TPCD.ORDERS.o_orderkey	
Condition 1 (Invariant)	TPCD.ORDERS.o_orderdate BETWEEN [1993-01-01 AND 1997-12-31]	
Condition 1 Selectivity	0.75952533	
Condition 1 Usefulness	3.24047487	
Condition 1 Distincts in Range	1826	
Condition 1 Elapsed time	0:00:00.125000	
Condition 1 Rows remaining after condition	1139288	
Condition 1 Index	FP(2) TPCD.ORDERS.ASIQ_IDX_T738_C5_FP	
Condition 1 range cost	FP: 0.8 HNG: 0.25 HG: 1.10038	
Condition 1 range cost	DATE:0.03	
Condition 1 Index Advisor	Add LF or HG index on TPCD.ORDERS.o_orderdate	

The important items to be checked in the leaf node are the following:

- 1. Table Row Count: count of rows actually stored in the table
- 2. Estimated Result Rows: expected row count after completion of leaf node execution
- 3. Condition K Selectivity: portion of the table that satisfied the predicate. This is displayed as a score between 0 and 1. A value approaching 0 is desirable.
- 4. Condition K Usefulness: the priority of predicate for execution flow. The predicate with the largest value is executed first, and the remaining predicates are executed in descending order of the usefulness value.
- 5. Condition K Index: the index used to process the condition. In the example above, the FP index for the column referenced in the predicate is used.

6. Condition K Index Advisor: if the optimizer determines that an LF or HG index is missing and would be useful, the advice is noted here. The optimizer does not provide advice on other types of indexes that might be useful and are missing.

5.1.3 Join Node Details

A join operation between two tables (left child node and right child node) is carried out in the join node. The optimizer determines the most efficient join algorithm from the information on data of the two input tables. Here is an example of a join node:

#04 Join (Sort-Merge)		
Parent Node	#33	
Left Child Node	<u>#36</u>	
Right Child Node	<u>#06</u>	
Generated Result Rows	1139288	
Estimated Result Rows	1139288	
Valid Join Algorithms	NLJ, SMJ, HJ	
Optimizer est. max hash rows for these keys	5000000	
Optimization Note	Optimized for primary key join	
Optimization Note	Replaced equated TPCD.CUSTOMER.c_custkey with TPCD.ORDERS.o_custkey	
Left Input Table 1	TPCD.ORDERS	
Estimated Left Inputs	1139288	
Right Input Table 1	TPCD.CUSTOMER	
Estimated Right Inputs	150000	
Join Result Constraint	Many to 1	

The important items to be checked in the join node are the following:

- 1. The join algorithm selected for the join (in the title box in this case: "Sort-Merge")
- 2. Left Child Node and Right Child Node: the leaf input nodes used for the join node
- 3. Estimated Result Rows: expected result row count after completion of execution in the join node
- 4. Valid Join Algorithms: algorithm candidates to execute the join (the actual join algorithm is shown in the join node identifier in the header in this case, a Sort-Merge join). The primary join candidates are:
 - a. Nested Loop Join (NLJ): loops through rows looking for a match on join key
 - b. Hash Join (HJ): probes into generated hash table with join keys
 - c. Sort Merge Join (SMJ): sorts, then merges join keys
- 5. Join Result Constraint: the relationship between the tables being joined: 1 to Many; Many to 1; Many to Many. A "Many to Many" relationship indicates that the optimizer cannot recognize how the tables relate to each other in a meaningful way, and you may need to add a primary key or unique index.

Of the various possible join algorithms, the hash type is the fastest.

5.1.4 Group Node Details

The GROUP BY statement in SQL is used in conjunction with an aggregate function to group the result set by one or more columns. For a GROUP BY, the optimizer must determine two things:

- 1. the expected number of resulting groups
- 2. which algorithm to use

To make these decisions, the optimizer uses a cost-based process that uses the available metadata for all columns used as grouping keys. There are two types of GROUP BY algorithms:

- Group By (Hash) for modest result sets
- Group By (Sort) for large sets

The "Group By (Hash)" algorithm tends to be the fastest.

Here is an example of a group by node that employs a "Group By (Sort)" algorithm:

#11 Group By (Sort)		
Parent Node	<u>#12</u>	
Child Node 1	<u>#25</u>	
Generated Result Rows	1139288	
Estimated Result Rows	4558074	
Dependent grouping expression (PK)	TPCD.CUSTOMER.c_name	
Dependent grouping expression (PK)	TPCD.ORDERS.o_totalprice	
Dependent grouping expression (PK)	TPCD.ORDERS.o_orderdate	
Optimizer est. max hash rows for these keys	869777	
Grouping Expression 1	FPVALUE(FPORDINAL(TPCD.ORDERS.o_custkey'(1), 3))	
Grouping Expression 2	TPCD.ORDERS.o_orderkey`(1)	
Non-Grouped Expression 1	TPCD.CUSTOMER.c_name'(1)	
Non-Grouped Expression 2	TPCD.ORDERS.o_totalprice'(1)	
Non-Grouped Expression 3	FPORDINAL(TPCD.ORDERS.o_orderdate*(1), 2)	
Aggregate Expression 1	SUM(FPVALUE(FPORDINAL(TPCD.LINEITEM.I_quantity'(1), 1)))	

6. What Steps Can You Take to Optimize a Query

When you suspect that a query is performing poorly, there are a few steps you can take to diagnose and fix the problem:

- 1. Evaluate your schema design: while schema design is important, SAP Sybase IQ tends to perform well with a variety of schema designs, and introducing a star or snowflake schema may not be required in order to have your queries perform well
- 2. Restructure your query: proper query design is important to generating the most efficient query
- 3. Add appropriate indexes to table columns involved in the query: SAP Sybase IQ makes heavy use of indexes when processing queries, both for determining metadata for proper execution flow and join algorithm selection, and for fast retrieval of data to satisfy the query. You are encouraged to create as many indexes as you need, because SAP Sybase IQ indexes are very efficient space-wise, and easy to manage.
- 4. Change database options to configure memory to bias towards a faster hash join or group by algorithm: it is important to have adequate memory to apply resources to satisfying the query in the fastest manner possible.
- 5. As with any RDBMS, there are hardware configuration improvements, such as multi-core chipsets and faster storage technologies, which can improve performance.

7. Actual Example of Improving a Query

In this example, we are going to optimize a query by turning on the index advisor, and adding a recommended index.

7.1 The query

This is the query we are going to optimize (the text for this query is in the file "\$TPCHROOT/TPCH/Queries/query20.sql"):

```
select
      s_name,
      s address
from
      TPCD.supplier,
      TPCD.nation
where
      s suppkey in (
            select
                   ps suppkey
            from
                   TPCD.partsupp
            where
                   ps_partkey in (
                         select
                               p partkey
                         from
                               TPCD.part
                         where
                               p name like 'peru%'
                   )
                   and ps availqty > (
                         select
                                0.5 * sum(l quantity)
                         from
                                TPCD.lineitem
                         where
                                l_partkey = ps_partkey
                               and 1 suppkey = ps suppkey
                               and l_{shipdate} >= "1996-01-01"
                               and 1 shipdate < '1997-01-01'
                   )
      and s nationkey = n nationkey
      and n name = 'FRANCE'
order by
      s name;
```

7.2 Turn on the Query Plan and the Index Advisor

In order to turn on query plan generation and the index advisor, you need to set some database options before running the query. From interactive SQL, execute the following commands (the text for these commands is in the file "\$TPCHROOT/TPCH/Metadata/SetOptions.sql"):

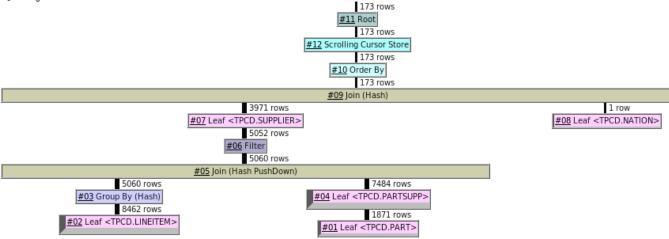
Note: Set the Query_Plan_As_HTML_Directory variable to an existing directory, or the HTML file will show up in the closest existing directory.

```
set option public.Query_Plan = 'ON';
set option public.Query_Detail = 'ON';
set option public.Query_Plan_After_Run = 'ON';
set option public.Query_Plan_As_HTML = 'ON';
set option public.Query_Plan_As_HTML_Directory = '/opt/sybase/TPCHDB/QueryPlans';
set option public.Query_Plan_Text_Access = 'ON';
set option public.Query_Plan_Text_Caching = 'ON';
set option public.Query_Timing = 'ON';
set option public.Index Advisor = 'ON';
```

When you do this, and execute the query, you will see an HTML file show up in the directory "/opt/sybase/TPCHDB/QueryPlans" (this directory was specified for the

"Query_Plan_As_HTML_Directory" database option above; you should choose an appropriate directory for your environment). This is the query plan in HTML format. Open up the file with a browser:

Query Tree



7.3 Index Advice

One of the features of the SAP Sybase IQ query plan is the Index Advisor. If you search for "Index Advisor" in the HTML file of the query plan, you will find multiple instances. These notations show where the SAP Sybase IQ Index Advisor is requesting an additional index, so that the query engine can execute the query more quickly. Here is one of the instances:

Time of Condition Done	2012-01-02 14:23:42.266435
Time of Delayed Condition Start	2012-01-02 14:23:42.982992
Time of Delayed Condition Done	2012-01-02 14:23:44.620604
Time of First Prepare	2012-01-02 14:23:42.563941
Time of First Prepare Return	2012-01-02 14:23:42.563950
Time of First Fetch Call	2012-01-02 14:23:42.982987
Time of First Fetch Return	2012-01-02 14:23:44.703056
Time of Complete Call	2012-01-02 14:23:45.757240
Time of Complete Return	2012-01-02 14:23:45.757530
Elapsed Condition time	0:00:00.656579
Elapsed time	0:00:04.284798
Condition 1 (Invariant)	TPCD.LINEITEM.I_shipdate BETWEEN [1996-01-01 AND 1997-01-01)
Condition 1 Selectivity	0.15221701
Condition 1 Usefulness	3.84778299
Condition 1 Distincts in Range	366
Condition 1 Elapsed time	0:00:00.656567
Condition 1 Rows remaining after condition	913487
Condition 1 Execution Method	Probe lookup table into bitmap cursor
Condition 1 Index	FP(2) TPCD.LINEITEM.ASIQ_IDX_T742_C11_FP
Condition 1 range cost	FP: 2.40049 HNG: 1.0002 HG: 1.1222
Condition 1 range cost	DATE:0.120024
Condition 1 Index Advisor	Add LF or HG index on TPCD.LINEITEM.I_shipdate
Condition 1 Number of Threads Used	5
Condition 1 Fragment ID	1
Condition 1 Find Work Units - tpch	20 (4, 4, 4, 4, 4)
Condition 1 Find Initial Work Unit Size	327680

The Index Advisor is requesting an HG index on LINEITEM.l_shipdate, because the column is used in a WHERE clause to filter down the number of returned rows. An HG index will help the query engine more quickly locate the LINEITEM rows that fall between the requested dates.

You can create an HG index on the column with the following SQL statement:

```
create HG index L_SHIPDATE_HG on TPCD.LINEITEM(L_SHIPDATE) in IQ USER MAIN;
```

Follow all the advice in the query plan to create the requested indexes and improve the query performance.

8. Summary

This chapter has shown you how to generate a query plan for a SAP Sybase IQ query. The query plan gives you quantitative metrics about query processing. You have also learned about the Index Advisor, and how to turn it on for clues about adding indexes to improve query performance.

Copyright

© Copyright 201H SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z10, System z9, z10, z9, iSeries, pSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects S.A. in the United States and in other countries. Business Objects is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.