



Business Objects Semantic Layer

Best Practices Accessing Databases

Contents

- INTRODUCTION 2**
- TUNING THE DATABASE 2**
 - Data models best practices 2*
 - Creating a new schema 2
 - Modifying an existing schema 4
 - Database server settings 4*
 - Oracle..... 5
 - Teradata..... 5
 - SQL Server 5
 - IBM DB2 5
 - Sybase..... 6
- TUNING THE CONNECTIVITY 6**
 - Increase the Array Fetch Size 6*
 - ConnectInit for optimization parameters..... 6*
- TUNING THE UNIVERSE 7**
 - Aggregate awareness 7*
 - Index awareness 7*
 - Merged joins (Join_By_SQL parameter) 8*
 - Derived tables 8*
 - Shortcut joins..... 9*
 - Table filtering based on size (Boundary_Weight_Table) 9*
 - Table filters in the FROM clause..... 9*
 - Multi-sourcing with Data Federator for optimized filtering 10*
 - Database analytic functions..... 11*
- TUNING THE REPORT 11**
 - Display all data in the same page or in a page by page mode 11*
 - Filter at query or at report level 11*
 - Choose the right scope of analysis 12*
 - Schedule reports 12*
- CONCLUSION 12**

Introduction

When it comes to optimization of a BI deployment and of the performance of queries on very large databases (VLDBs), we find that each situation has specific issues and specific solutions. The Business Objects semantic layer doesn't force the customer to adopt a dedicated database architecture or schema and it allows for personalized optimization settings while adapting to any data model and source type. Optimizations can be added at different levels: database, connectivity, universe, and report. Improvements and additional tunings can be done during the whole lifecycle of the BI solution with a minimal or even null impact on the final user workflows (but for an improved performance). In this document, we will discuss possible improvements which can be done in a short period. The list is not exhaustive and technical details are left out.

Tuning the database

The source database can be tuned in many ways to achieve a better query performance. Some tuning on the metadata model is common to all database servers; other tuning is server-specific. As a rule of thumb, letting the database do the calculations or even pre-calculating values in the database leads to better query performance. Pre-calculations and other optimizations require knowing in advance the kind of queries which will be executed.

As this is quite a complex task with our ad-hoc query tools, we give the opportunity to tag a query as coming from one of our applications. This allows the database administrator to analyze our queries and decide what database optimizations can be made.

One of the benefits of the Business Objects semantic layer is that it can access any data schema and doesn't require the creation of new tables, views, or indexes in the database. This said, if a user has the opportunity to modify the database model or create a new one to be used by Business Objects applications, than this model can be made to suit the best practices for optimal query performance and can be subsequently fine-tuned.

Data models best practices

We often find schemas already configured with few possibilities for modification. In this case we suggest changes as less intrusive as possible to increase performance.

Creating a new schema

Although Business Objects semantic layer can be used on any database schema, organizations often create new models in concert with the deployment of our tools. The following sections highlight some possible

optimizations which can be done at the schema level for use with Business Objects applications.

As Business Objects tools allow for easy ad-hoc reporting, the optimizations might have to change in time to adapt to new business requirements. Those changes can be done in a transparent way for the users.

Limit the number of joins while keeping the record size low

If the queries are known in advance and there is complete freedom in the configuration of the data model, the schema should be fine-tuned to decrease the number of joins and the quantity of data which is necessary for the generation of a correct answer but which is not needed in the final answer.

Joins can be reduced by putting in the same table the items which are supposed to be requested often together or by creating join indexes or tables to bypass the normal whole join path from one data to another.

A decrease of data can be obtained by pre aggregating values in specific tables, by creating views or by creating pre-filtered set of data in other tables.

Star schema has been tested as the fastest responding schema

By benchmarking various schemas against typical queries, we found that the Star Schema is generally the most performing one. If the query sentences are hard to define because users will do a lot of ad-hoc reports or will do complex analysis using different exploration paths, the Star Schema usually provides a very good default for obtaining performing queries.

Aggregate tables

Aggregation of data within tables avoids complex calculations at the Business Objects semantic layer level. The semantic layer offers a functionality called Aggregate Awareness which automatically makes the SQL sentence optimally point to the highest aggregation level available in the schema for the requested result set.

Partitioned tables

Table partitioning is a database feature allowing a table to be split in many parts, each one with a specific partitioning dimension (e.g. split a Sales tables per Year of Sale). When receiving the query, the database will send the request to the correct table without the need to parse all data.

When building a schema for Business Objects applications, if typical queries are known, it will be possible to do a partitioning based on the most used filters to speed up the query performance.

Create Indexes based on workload

Indexes improve greatly the performance of queries making use of them. There are many types of Indexes with different approaches for

improving the performance of a query. The choice of indexes can be done when designing the schema based on the estimated requests which will be done by the Business Objects tools.

Many database vendors provide tools for optimizing the indexes which audit the SQL to be processed and suggest possible indexes to be created. Business Objects tools provide ways to retrieve the final SQL generated and passed to the database. By running the SQL in the optimization tools, it is possible to see what the best indexes to create are.

Modifying an existing schema

In many situations, a schema already exists at the customer site and it is not possible to modify it because other applications make use of it. Even if the existing schema cannot be changed, some enhancements can be added and other items can be redefined in a way to keep the existing experience for users. Below are some enhancements which can be done to an existing schema.

Aggregate tables

As discussed above, aggregate tables can provide better performance by pre-calculating values at the database level. Aggregate tables can be added to an existing schema with no impact for existing applications. The addition of aggregation tables requires additional space into the database, but this space is not always large and it can be pre-calculated (and limited with some DB-specific tools) in advance.

Database Views

If space is an issue, database views can be created to aggregate or pre-select information stored in tables. New database views have no impact on existing applications already accessing the database. Views are less performing than aggregate tables as the data is to be computed each time they are accessed, but do not require updates or take up significant space in the database. Calculations done at database view level might be faster than the same calculations done at the Business Objects client tool or universe level. Business Objects semantic layer treats views as if they were tables in the database.

Create new indexes based on existing workload

As described in the previous section, Indexes can dramatically improve performance of a query. In many cases it is possible to add indexes to a database without impact for existing applications and without the need for modifying what already exist in the database schema. Business Objects tools provide a way to track what SQL sentence is generated by them. By running the index optimization tools provided by the database vendor on the report' SQL, it is possible to create new indexes at any time during the BI deployment lifecycle.

Database server settings

The performance of a query is often strictly linked to a good tuning of the database server parameters and the database itself. Below is a list of

database-specific features that can be taken into account when optimizing a database for use with Business Objects.

Oracle

The following features can be considered:

- Star-schema optimizer
- Materialized views
- Table compression

Additional information can be found in the following document

http://www.oracle.com/technology/deploy/performance/pdf/twp_perf_database%20performance%20with%20oracle10gr2.pdf

Teradata

The following features can be considered:

- Define primary indexes on most used filters
- Run Index wizard on Business Objects' queries (use END_SQL to find them)
- Create Join Indexes

SQL Server

The following features can be considered:

- Use Table and Index partitioning

Additional information provided by the vendor can be found in the following documents

<http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/rdbmspart.msp>

http://www.microsoft.com/technet/prodtechnol/sql/bestpractice/dw_perf_top10.msp

IBM DB2

The following features can be considered:

- Use Cube Views on the data schema accessed by a universe
- Create a Cube Views from a universe using the Business Objects Metadata bridge
- Managed Query Tables

Additional information provided by the vendor can be found in the following document:

<http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0404mcarthur/>

Sybase

The following features can be considered:

- In Sybase ASE set the session optimization to DSS

Tuning the Connectivity

The connectivity is the communication layer between the data source and our tools.

Making sure the connectivity is properly configured is the first step towards good query performance. Some connectivity parameters can influence the final performance of the query and can be changed. Any change from the default of these parameters must be done with care as it will greatly influence the way our tools talk to the database. We suggest to perform tests from Universe Designer and the query and reporting tools (Web Intelligence, Desktop Intelligence, Crystal Reports) when modifying these values to make sure that there is a real performance improvement.

Increase the Array Fetch Size

The Array Fetch Size defines how many rows are returned from the database at each fetch. If the network can support large packets it is possible to increase the size of the Array Fetch Size to increase the number of lines retrieved each time and decrease the number of fetch operations.

NOTE	If you need to retrieve 100 rows and have an Array Fetch Size of 20, then 5 fetch operations will be executed. If you have the value to 100 or higher, then only 1 fetch operation will be executed (but the network package will be bigger).
-------------	---

It is useful to test with different values to find the correct balance of fetch operations vs. size of network transfers.

ConnectInit for optimization parameters

Using the ConnectInit parameter it is possible to send commands to the database when opening the session which can be used to set database-specific parameters used for optimization.

NOTE	<p>With Sybase ASE it would be possible e.g. to set the database query optimization algorithm to perform DSS queries and not the default OLTP ones for the session.</p> <p>With Teradata it would be possible e.g. to set the row read lock level to access for the session to generate an optimized tactical query</p> <pre>SET SESSION CHARACTERISTICS AS TRANSACTION ISOLATION LEVEL RU</pre>
-------------	--

Tuning the Universe

Some automatic SQL improvements are available by default in the semantic layer query generator: sorting the tables in order of size when joining, searching for the shortest join path, filtering in the table which minimizes the number of joins. Below is a list of other tuning techniques can be used while designing universes.

Aggregate awareness

If Aggregate tables are available in the database, Business Objects semantic layer offers a specific function to transparently make use of them.

With the @Aggregate_Aware function, a measure can have multiple definitions across multiple tables. At query time, based on the level of detail requested by the final user, the SQL generator will create a SQL query using the highest possible level of aggregation. Using aggregate tables guarantees a faster query result because those tables are smaller than detailed fact tables and that no additional calculation is needed either in the database or in Business Objects tools.

EXAMPLE

In a data model there is a fact table for sales on a transaction level detail and an aggregated table with sales per day. If the final user chooses to see sales details then the transaction table will be used, if the user wants to see sales per day then the aggregate table will be used. This is fully transparent to the final user.

Index awareness

It is possible to have the query generator to make use of keys instead of labels whenever possible. This feature optimizes the SQL as it decreases the number of joins needed in a sentence and parses (usually ordered) numbers instead than text. Just by defining the primary and foreign keys for an object, the SQL will take care of generating the most performing SQL.

NOTE

In a Country/Region/City/Customer hierarchy, the use of Index Awareness can produce the following results on a query for Customer and Region filtered on Country=France

Index Awareness

Without

```

SELECT
  Region.region,
  Customer.last_name
FROM
  Country INNER JOIN Region ON
  (Country.country_id=Region.country_id)
  INNER JOIN City ON
  (City.region_id=Region.region_id)
  INNER JOIN Customer ON
  (City.city_id=Customer.city_id)

WHERE
  Country.country = 'France'
          
```

4 tables
3 joins
Where on label

With

```

SELECT
  Region.region,
  Customer.last_name
FROM
  Customer INNER JOIN City ON
  (City.city_id=Customer.city_id)
  INNER JOIN Region ON
  (City.region_id=Region.region_id)

WHERE
  Region.country_id = 2
          
```

3 tables
2 joins
Where on key

Slide 51

Copyright © 2007 Business Objects S.A. All rights reserved.



Merged joins (Join_By_SQL parameter)

When requesting measures coming from two different contexts (two different fact tables sharing the same dimensions), the Business Objects semantic layer generates two SQL sentences and joins the results within our tools. It is possible to force the joining of the two result sets at the database level. Forcing it into the database provides performance improvement as the calculation power of the database might be stronger on its own data.

To enable this feature, the universe parameters must contain the value `Join_By_SQL=Yes`.

Derived tables

Some calculations or filtering which are done at universe or report level could be pushed down to database level (without the need to actually modify the database). This has two advantages: final users have less work to do (they do not need to create the calculation); there is an improvement of performance as the database calculation engine might be faster on the database data and there is less data transferred.

To push down calculations at database level, in the universe, we can define Derived Tables. Derived Tables are SQL sentences which define a virtual table (but do not actually create any table in the database). Designers will see those Derived Tables as plain tables in the database. At query time, if such tables are needed, the SQL sentence defining them will be added to the query.

NOTE

In Teradata, Derived Tables have a better performance than Virtual Tables. We do not support Virtual tables since their functionality can be obtained by using the more performing and dynamic Derived Tables.

Shortcut joins

Shortcut Joins are alternative join paths which designers can define in the data foundation wherever possible. The query generation engine, when defining the query sentence, will look for all possible paths and will choose the shortest one in term of joins. Shortcuts are not taken into account to define contexts but only to decrease the number of joins whenever possible.

In this data foundation, you can go from Amount_Sold to Promotion_cost directly without the need to join the Article_lookup table. The SQL generator will decide the fastest path based on the objects requested in the query.

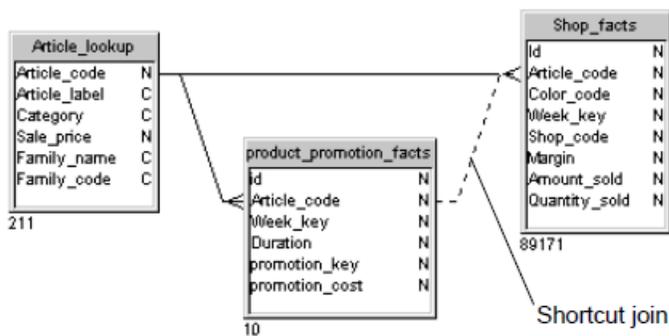


Table filtering based on size (Boundary_Weight_Table)

Large tables in the From clause have to be wholly parsed even if they are filtered by a condition afterward. By setting the Boundary_Weight_Table parameter, designers can define a limit (in number of rows) to avoid this full parse. If the table size in rows is greater than the entered value, the table is declared as a subquery:

```
FROM (SELECT col1, col2, ..., coln FROM Table_Name WHERE
simple condition)
```

In the statement above, only the columns necessary for the query are retrieved and the condition in the inner statement limits the number of records to parse in the outer one.

Table filters in the FROM clause

Independently of the table size (as per the section above), it is possible to define that all (or a subset of) objects cause the filters (which usually appear in the Where clause) to be moved up to the FROM clause. By moving filters in the FROM clause the query will parse a smaller set of data.

A statement like

```

...
FROM
    Country INNER JOIN Region ON
    (Country.country_id=Region.country_id)
...
WHERE
    Country.country In ( 'France', 'US' )

```

Becomes

```

...
FROM
    Country INNER JOIN Region ON
    (Country.country_id=Region.country_id AND Country.country
    In ( 'France', 'US' ))
...

```

In the first statement the join is done on all IDs and then the data is filtered, in the second statement, the data is filtered and then the join is done on all remaining IDs.

Multi-sourcing with Data Federator for optimized filtering

In some scenarios, users might need to retrieve data from multiple sources and use it into a report. Business Objects Web Intelligence provides the feature 'Multi Data Provider Synchronization' to merge multi-source data at the report level.

It is also possible to push the data federation down at the universe level using Business Objects Data Federator. This provides a couple of benefits: the user does not have to deal with the two different data sources and the query performance is improved thanks to the Data Federator intelligent generation. The benefit of using the Data Federation multisource engine is dramatic when filtering a very large database table with a short list of values from elsewhere.

EXAMPLE	<p>A user needs to retrieve all sales for a list of 15 customers. The sales information comes from the data warehouse, the list of customers comes from a small Access database.</p> <p>One way to generate the query is to retrieve all customers from Access, all Sales from the warehouse and then filter within our own tools.</p> <p>Business Objects Data Federator reads the list of customers from Access first, then queries the warehouse for all sales where the customer is in the list. The data retrieved from the warehouse is smaller than in the first case where all sales were returned.</p>
----------------	---

Database analytic functions

Many database vendors provide a full set of analytic functions which give OLAP-like functionality within the relational framework. Those functions are quite powerful as they allow calculating in the database itself. The calculation at database level can be faster as it is done on data within the source itself and because less information is sent back to the query / reporting application.

NOTE	<p>Analytic functions usually include Ranking, Accumulative aggregation, Ratio, Ratio to Report, and Reporting Aggregate. Thanks to the open semantic layer architecture of Business Objects, it is possible to use any function available in the database.</p> <p>Some examples of such functions are:</p> <p>RANK (over partition), SUM (over partition), Ratio_to_Report (over partition) for DB2 or Oracle;</p> <p>RANK, CSUM, MAVG, MDIFF, MLINREG, MSUM for Teradata</p>
-------------	--

Tuning the report

Reports are the final destination of data within our solution. It is possible to create reports taking into account performance.

Display all data in the same page or in a page by page mode

When viewing a report within Web Intelligence, you can choose to see all data in the same page or in a preformatted page by page mode. Reading data page by page only download from the server the data necessary to render the current page. With very large quantity of data, you can choose if waiting more at the beginning and getting all the data to browse at a later stage (with no delay), or waiting less to see only the current page but repeat the wait each time the page is changed.

Filter at query or at report level

When creating a report in Web Intelligence you can decide to filter while creating the query (the SQL sent to the database will contain the filter) or filter the data afterwards when it has been put in the reports.

By filtering in the query, you will retrieve less data from the database, hence the query will be faster. This technique is best suited if you are confident that you will get all the data needed for your analysis without the need to go back to the database.

By filtering in the report, you will retrieve more data than needed from the database, hence the query will be slower. This technique is best suited if you are not sure of the level of details needed from the database to complete your analysis. More time will be needed to run the query but afterwards, by filtering within the report, no additional time will be required to go back to query the database.

Choose the right scope of analysis

Our query panel allows retrieving different levels of detail by setting the scope of analysis. As an example if you have a geography hierarchy and you select 'Country', the system also retrieves 'State' and 'City' if you set the scope of analysis to go two levels down.

To guarantee a good performance make sure to retrieve only the needed scope of analysis. Similarly as discussed for the filters, reduce the scope of analysis if you are confident that you have all needed data and that you won't need to go back to the database. Increase the scope if you are not sure of the level of details needed so that you can stay within the report for your analysis.

Schedule reports

If a report takes a long time to refresh, it might be best to schedule it and just open it without refresh once it is available. Our tools can notify users when a report is ready to be read after a refresh. To allow for a deep analysis within the report without the need to go back to the database, the report can be set with a large scope of analysis and a reduced number of filters, and be read in a one-page only mode. Once the report is done, the exchanges with the database will be null and the exchanges with the Web Intelligence server will be minimized while keeping full interactivity and analysis capability.

Conclusion

The Business Objects semantic layer can adapt to any database size and any database schema. It allows tuning for performance at all levels of the data chain, from the database server settings to the final report. Whenever possible, the Business Objects semantic layer pushes the load to the database to make the best use of the database capabilities and resources. The 'isolation' level provided by the semantic layer allows for heavy database- or universe-level changes during the lifetime of an evolving BI solution without any impact for the end users' work (apart from an improved performance). A good performance of the business intelligence solution starts with a well planned project where final users, BI experts and database administrators work together to define requirements and optimizations needed at all levels.

► www.businessobjects.com

No part of the computer software or this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from Business Objects.

The information in this document is subject to change without notice. Business Objects does not warrant that this document is error free.

This software and documentation is commercial computer software under Federal Acquisition regulations, and is provided only under the Restricted Rights of the Federal Acquisition Regulations applicable to commercial computer software provided at private expense. The use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in subdivision (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at 252.227-7013.

The Business Objects product and technology are protected by US patent numbers 5,555,403; 6,247,008; 6,578,027; 6,490,593; and 6,289,352. The Business Objects logo, the Business Objects tagline, BusinessObjects, BusinessObjects Broadcast Agent, BusinessQuery, Crystal Analysis, Crystal Analysis Holos, Crystal Applications, Crystal Enterprise, Crystal Info, Crystal Reports, Rapid Mart, and WebIntelligence are trademarks or registered trademarks of Business Objects SA in the United States and/or other countries. Various product and service names referenced herein may be trademarks of Business Objects SA. All other company, product, or brand names mentioned herein, may be the trademarks of their respective owners. Specifications subject to change without notice. Not responsible for errors or omissions.

Copyright © 2007 Business Objects SA. All rights reserved.