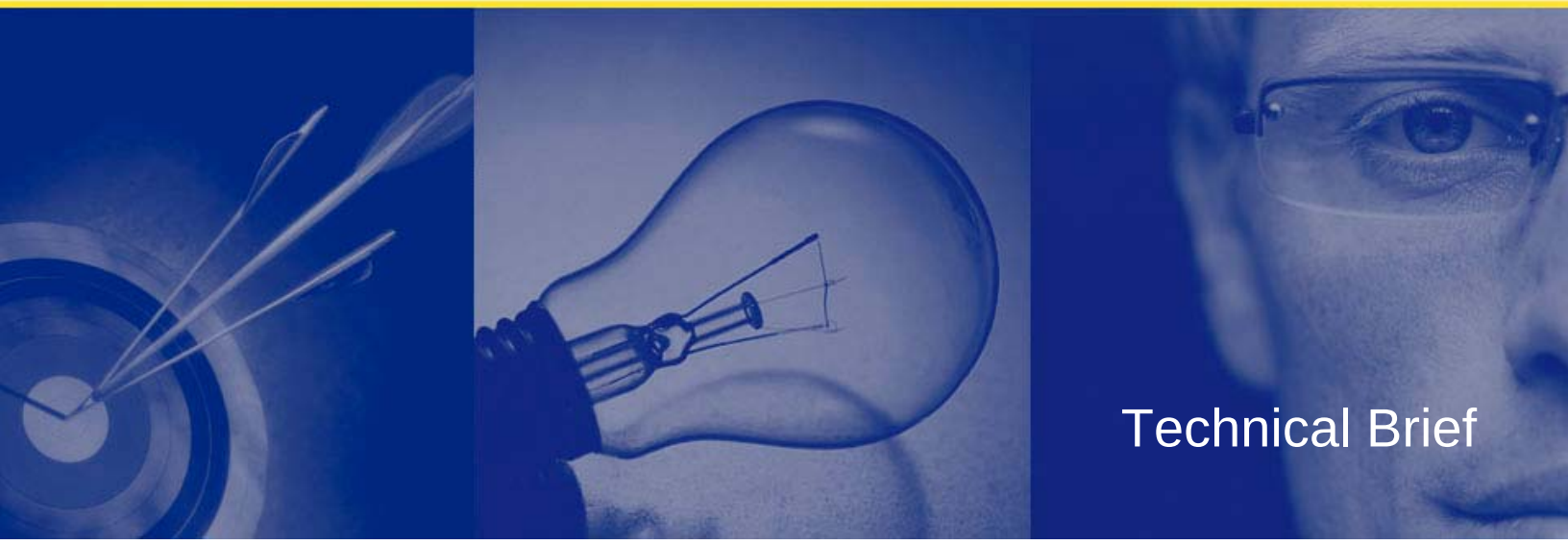
QUEST SOFTWARE®

The Value of Reverse Engineering

written by
Bert Scalzo, Ph.D.
Product Architect
Quest Software, Inc.



World Headquarters
5 Polaris Way
Aliso Viejo, CA 92656
www.quest.com
e-mail: info@quest.com
Please refer to our Web site for regional and international office information.
Updated—January 26, 2007

Content

Introduction

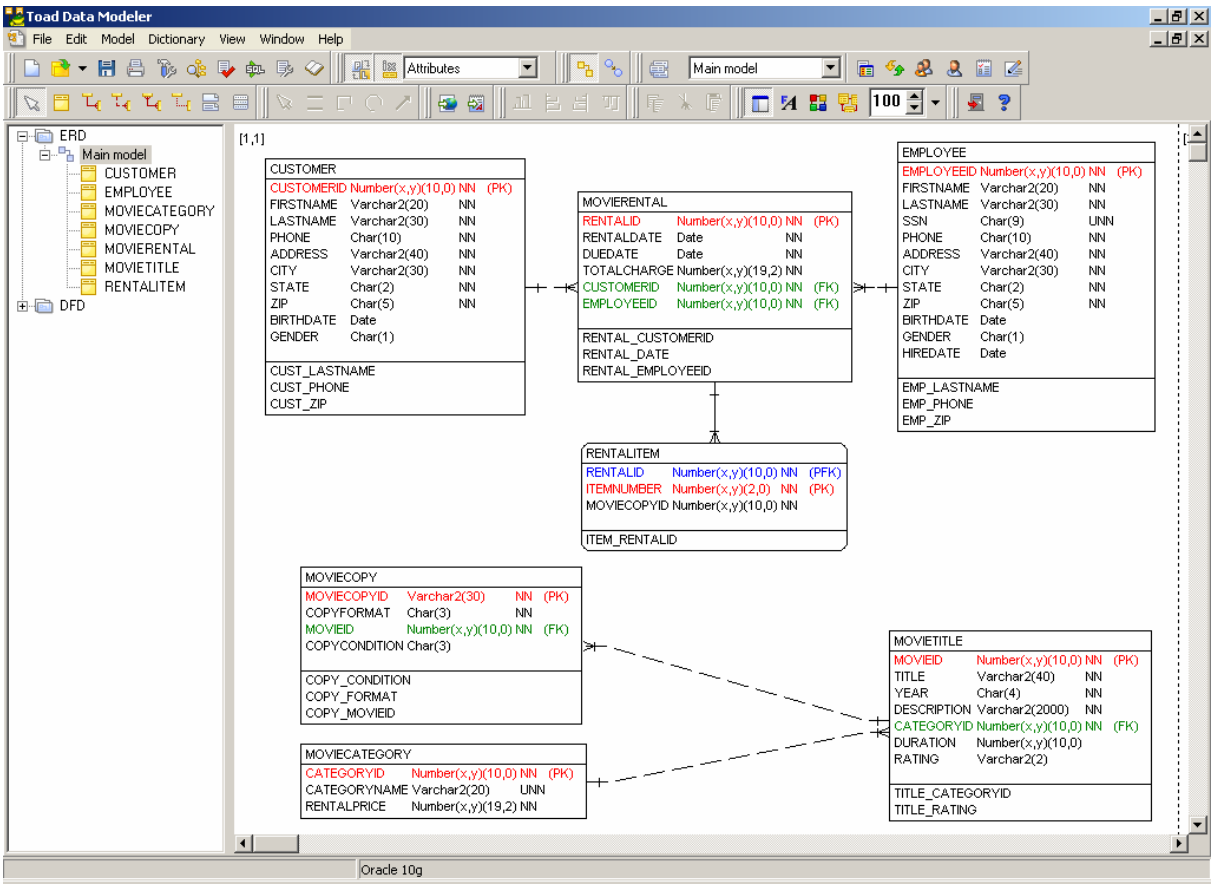
In many shops that I visit, neither the DBAs nor the database developers attribute much value to performing data modeling. And often, there are seemingly sound reasons for this exclusion. First, the steep price of data modeling tools is cost-prohibitive. Second, data models are not required as a project milestone or a deliverable. Third, the developers cannot modify a database because they've either inherited it from someone else or it supports a third-party application.

Addressing the Issues

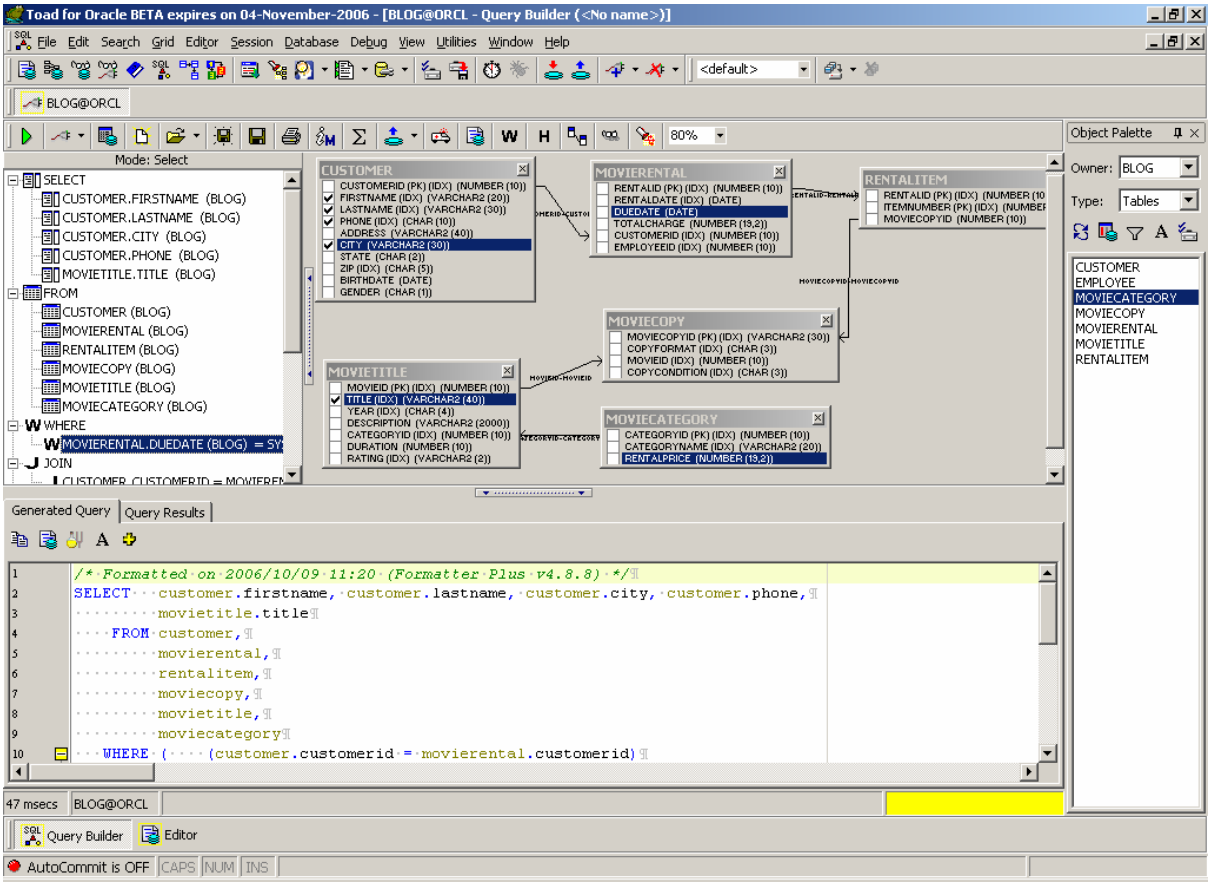
Quest's new Toad Data Modeler resolves some of these issues. At just $479, the data modeler is approximately one-sixth the cost of comparable tools. The second issue will not be addressed since it's tied to the maturity of the application development life cycle. However, I would like to focus on the third reason. Database design is static, which demonstrates that reverse engineering can still yield significant value.

Building Queries

A simple data model of a basic video rental system is shown below.



First, take a look at the challenges from the database developer's viewpoint. Suppose we are assigned to create a report of the customer name, city, phone number, movie title and rental price for all movies due back today. Imagine writing a query without an image similar to the one above. Most of us might revert to hand-drawing a rough diagram. Another alternative would be to utilize Toad for Oracle's graphical query builder shown below to visually construct the query. Should the first attempt need improvement, the SQL Editor can further refine the diagram.



The SQL statement is easy to comprehend. The following condensed query provides readability:

```
SELECT  customer.firstname, customer.lastname, customer.city,
        customer.phone, movietitle.title
  FROM  customer, movierental, rentalitem, moviecopy,
        movietitle, moviecategory
 WHERE customer.customerid = movierental.customerid
   AND movierental.rentalid = rentalitem.rentalid
   AND rentalitem.moviecopyid = moviecopy.moviecopyid
   AND movietitle.movieid = moviecopy.movieid
   AND moviecategory.categoryid = movietitle.categoryid
   AND movierental.duedate = SYSDATE
ORDER BY customer.city ASC, moviecategory.rentalprice ASC
```

The database developer may think the report is complete. However, the query offers two full table scans (highlighted in red), and the run time of 46 ms can be improved significantly.

Plan
SELECT STATEMENT ALL_ROWS Cost: 17 Bytes: 236 Cardinality: 1
16 SORT ORDER BY Cost: 17 Bytes: 236 Cardinality: 1
15 NESTED LOOPS Cost: 16 Bytes: 236 Cardinality: 1
12 NESTED LOOPS Cost: 15 Bytes: 210 Cardinality: 1
9 HASH JOIN Cost: 14 Bytes: 162 Cardinality: 1
7 NESTED LOOPS Cost: 10 Bytes: 132 Cardinality: 1
4 NESTED LOOPS Cost: 9 Bytes: 61 Cardinality: 1
1 TABLE ACCESS FULL TABLE BLOG.MOVIERENTAL Cost: 8 Bytes: 35 Cardinality: 1
3 TABLE ACCESS BY INDEX ROWID TABLE BLOG.RENTALITEM Cost: 1 Bytes: 52 Cardinality: 2
2 INDEX RANGE SCAN INDEX BLOG.ITEM_RENTALID Cost: 1 Cardinality: 2
6 TABLE ACCESS BY INDEX ROWID TABLE BLOG.CUSTOMER Cost: 1 Bytes: 71 Cardinality: 1
5 INDEX UNIQUE SCAN INDEX (UNIQUE) BLOG.CUSTOMER_PK Cost: 1 Cardinality: 1
8 TABLE ACCESS FULL TABLE BLOG.MOVIECOPY Cost: 3 Bytes: 24,000 Cardinality: 800
11 TABLE ACCESS BY INDEX ROWID TABLE BLOG.MOVIETITLE Cost: 1 Bytes: 48 Cardinality: 1
10 INDEX UNIQUE SCAN INDEX (UNIQUE) BLOG.MOVIETITLE_PK Cost: 1 Cardinality: 1
14 TABLE ACCESS BY INDEX ROWID TABLE BLOG.MOVIECATEGORY Cost: 1 Bytes: 26 Cardinality: 1
13 INDEX UNIQUE SCAN INDEX (UNIQUE) BLOG.MOVIECATEGORY_PK Cost: 1 Cardinality: 1

On the other hand, the DBA may address the performance and tuning issues. The data model actually holds some important clues. First, notice the missing relationship line between RENTALITEM and MOVIECOPY. The foreign key constraint is not defined and the data lacks the MOVIECOPYID index in the largest RENTALITEM table for its merge with MOVIECOPY.

Furthermore, notice the difference on the data type for MOVIECOPYID across the two tables. Even with the creation of an index on the RENTALITEM column, the explain plan would have to perform implicit data type conversions during the join. Therefore, two big performance problems are hiding from the obvious exclusion.

Second, observe how the query has a WHERE clause restriction that references DUEDATE on the second largest table of MOVIERENTAL. Once again, the data model omits the index in the column. Finally, the ORDER BY clause on two non-indexed columns (CITY and RENTALPRICE) appears. Although this last issue does not directly affect the data model, the sort order will be used often. Thus, it would make sense to index these columns.

Reverse Engineering

By fixing the simple and obvious design flaws, the result exposes an improved explain plan (with no full table scans) – not to mention a meager 16 ms runtime. The brief review of the data model contributes to a 65 percent improvement.

Plan
SELECT STATEMENT ALL_ROWS Cost: 7 Bytes: 232 Cardinality: 1
18 SORT ORDER BY Cost: 7 Bytes: 232 Cardinality: 1
17 NESTED LOOPS Cost: 6 Bytes: 232 Cardinality: 1
14 NESTED LOOPS Cost: 5 Bytes: 206 Cardinality: 1
11 NESTED LOOPS Cost: 4 Bytes: 158 Cardinality: 1
8 NESTED LOOPS Cost: 3 Bytes: 132 Cardinality: 1
5 NESTED LOOPS Cost: 2 Bytes: 106 Cardinality: 1
2 TABLE ACCES BY INDEX ROWID TABLE BLOG.MOVIERENTAL Cost: 1 Bytes: 35 Cardinality: 1
1 INDEX RANGE SCAN INDEX BLOG.DUE_DATE Cost: 1 Cardinality: 1
4 TABLE ACCESS BY INDEX ROWID TABLE BLOG.CUSTOMER Cost: 1 Bytes: 71 Cardinality: 1
3 INDEX UNIQUE SCAN INDEX (UNIQUE) BLOG.CUSTOMER_PK Cost: 1 Cardinality: 1
7 TABLE ACCESS BY INDEX ROWID TABLE BLOG.RENTALITEM Cost: 1 Bytes: 52 Cardinality: 2
6 INDEX RANGE SCAN INDEX BLOG.ITEM_RENTALID Cost: 1 Cardinality: 2
10 TABLE ACCESS BY INDEX ROWID TABLE BLOG.MOVIECOPY Cost: 1 Bytes: 26 Cardinality: 1
9 INDEX UNIQUE SCAN INDEX (UNIQUE) BLOG.MOVIECOPY_PK Cost: 1 Cardinality: 1
13 TABLE ACCESS BY INDEX ROWID TABLE BLOG.MOVIETITLE Cost: 1 Bytes: 48 Cardinality: 1
12 INDEX UNIQUE SCAN INDEX (UNIQUE) BLOG.MOVIETITLE_PK Cost: 1 Cardinality: 1
16 TABLE ACCESS BY INDEX ROWID TABLE BLOG.MOVIECATEGORY Cost: 1 Bytes: 26 Cardinality: 1
15 INDEX UNIQUE SCAN INDEX (UNIQUE) BLOG.MOVIECATEGORY_PK Cost: 1 Cardinality: 1

Finally, if the database was reverse-engineered and the modeling tool was used to find and correct such issues, a model verification report can be processed. The convenient and easy verification process will often expose many harmless flaws and result in optimal database performance.

About the Author

Bert Scalzo is a Product Architect for Quest Software and a member of the TOAD team. He has worked extensively with TOAD's developers and designed many of TOAD's features, including the DBA module, Code Road Map and Code Xpert. Mr. Scalzo has more than two decades of experience with Oracle databases, starting with version 4, and served at both the Oracle Education and Oracle Consulting organizations. He holds several Oracle Masters certifications as well as bachelor's, master's and Ph.D. degrees in computer science. Additionally, Mr. Scalzo holds an MBA and several insurance industry designations. He is also an accomplished speaker and has presented at numerous Oracle conferences and user groups. He is the author of four books: "Oracle DBA Guide to Data Warehousing and Star Schemas", "TOAD Handbook", "TOAD Pocket Reference" (2nd Edition), and "Database Benchmarking: Practical Methods for Oracle 10g & SQL Server 2005". You can reach Mr. Scalzo at bert.scalzo@quest.com or bert.scalzo@yahoo.com.

About Quest Software, Inc.

Quest Software, Inc. delivers innovative products that help organizations get more performance and productivity from their applications, databases and Windows infrastructure. Through a deep expertise in IT operations and a continued focus on what works best, Quest helps more than 50,000 customers worldwide meet higher expectations for enterprise IT. Quest Software can be found in offices around the globe and at www.quest.com.

Contacting Quest Software

Phone:	949.754.8000 (United States and Canada)
Email:	info@quest.com
Mail:	Quest Software, Inc. World Headquarters 5 Polaris Way Aliso Viejo, CA 92656 USA
Web site:	www.quest.com

Please refer to our Web site for regional and international office information.

Contacting Support

Quest Support is available to customers who have a trial version of a Quest product or who have purchased a commercial version and have a valid maintenance contract. Quest Support provides around the clock coverage with SupportLink, our web self-service. Visit SupportLink at http://support.quest.com

From SupportLink, you can do the following:

- Quickly find thousands of solutions (Knowledgebase articles/documents).
- Download patches and upgrades.
- Seek help from a Support engineer.
- Log and update your case, and check its status.

View the ***Global Support Guide*** for a detailed explanation of support programs, online services, contact information, and policy and procedures. The guide is available at: http://support.quest.com/pdfs/Global Support Guide.pdf