EMBARCADERO TECHNOLOGIES®

Ringing the Changes for Change Management

Philip Rathle and Scott Walz

Originally published in *ISUG Technical Journal*, November/December 2007
www.isug.com

Corporate Headquarters
100 California Street, 12th Floor
San Francisco, California 94111

EMEA Headquarters
York House
18 York Road
Maidenhead, Berkshire
SL6 1SF, United Kingdom

Asia-Pacific Headquarters
Level 9, 390 St Kilda Road
Melbourne VIC 3004
Australia

Ringing the changes for change management

By Philip Rathle and Scott Walz

As modern database change management techniques for the Sybase DBA become more sophisticated, there is an inherent need for practitioners to be aware of the increasingly powerful tools available to them. Philip Rathle and Scott Walz of Embarcadero rifle through their DBA toolkit and sharpen up a few blades for us.

Philip Rathle brings over ten years of experience with mission critical systems in the areas of customer data management, data warehousing, marketing and campaign management and online operational data management to his role as Principal Consultant at Embarcadero.

Scott Walz is a Senior Product Manager at Embarcadero Technologies, he oversees the direction of the company's database development products as well as database research and development for the engineering departments. He can be reached at scott.walz@embarcadero.com

As mature and pervasive as Sybase ASE has become, nowhere does there appear to have emerged a universally-accepted system or standard that addresses the problem of database change management in an holistic fashion. DBA's sit at the center of a complex dance that touches many participants: data modelers, database developers, architects, business analysts, software developers and more.

At any given time, each database is physically instantiated across a number of different environments, each of which may contain any one of several versions of the same database. Furthermore, the design of a particular database is typically stored across a variety of locations and a multiplicity of tools, which may include a data modeling tool, a SQL development tool, a database administration tool, a database change management tool and so on.

Shepherding the stream

Accurately shepherding the stream of functional and technology change across all of these physical environments and design layer components can be a tremendous challenge. Oftentimes, the database change process has been (rightly or wrongly) patterned after the software change management process. This invariably leaves gaps, which DBA's usually fill with manual and ad hoc workarounds. The job gets done, however the process for achieving it is generally not optimal for the specifics of database applications. Moreover, it often comes at a cost of a gradually diverging design layer, as this layer lies for the most part outside of the DBA's purview.

DBA's sit at the center of a complex dance that touches many participants: data modelers, database developers, architects, business analysts, software developers and more.

The good news is that there is some new hope for this old issue. The emergence of a powerful new breed of database change management tools has breathed some new life into this space and will begin yielding some relief. While tools can never of themselves be silver bullets for complicated process problems, their adoption is an important element not just in the solution, but in instigating change.

Today's enterprises are so large and complex that one cannot hope to climb out of today's convoluted database change management ruts without new and powerful functionality aimed specifically at this problem. Database change management tools, and the techniques they enable, play an important role not merely in coordinating the overall change management process, but in unwinding and assessing the current state of affairs so that it can make the leap forward.

Some of the key difficulties in managing database change are:

- the need to preserve data when making a structural change
- the need to maintain separate storage settings for objects across environments–even when the structures are themselves identical
- security differences between environments, reflected in users, roles and object permissions
- shared responsibility for certain types of objects, for example stored procedures, which are code artifacts and at the same time database artifacts and may be "owned" either by the DBA or the database developer, depending on the organization and on the target environment
- the need to keep physical and logical models in sync with the database (which can itself have cascading impacts)
- the need to manage and (increasingly) report on differences between database settings over time within a database
- the need to manage and maintain different sets of database settings across different database environments
- the need to validate the synchronicity of data in a replicated environment by comparing data between primary and replicated tables
- the need to manage reference data as one of the components necessary for a database build, complicated by the fact that reference data can be represented as table data or as check constraints, and is often shared across loosely-related databases
- the need to support multiple versions of a database concurrently, to support parallel branches of development
- the need to accommodate multiple paths by which change may be effected. For example, emergency changes in the middle of the night that will nearly always be made by executing a DDL directly against the database and not through a change management framework (let alone a data modeling tool!) Yet these changes ultimately need to be reflected in all of these places.

Documentation duties

If it weren't enough to get the right structure of data into the right place with as little downtime as possible, what is becoming equally crucial for DBA's is documenting what happened, when, and by whom.

These challenges are well understood by database professionals, who deal with them on a day-to-day basis. However they tend not to be fully appreciated by those who are not so closely associated with the technology. Solutions to these problems tend to diverge widely from organization to organization and to rely heavily on manual processes. This is a testament to the real lack of visibility which this important issue suffers.

Part of the problem is that very few organizations look at "Database Change Management" as a single unified process. It is assumed that database change management is a natural outcome, at the juncture of database management, software configuration management, and data modeling. The first step in developing a robust database change management solution is to recognize it as a standalone process or discipline, with its own peculiarities that make it quite different from software change management.

One must then ask the question, "What is database change management?" We believe that the question should consider not just structures, but also settings and data. These are the three primary factors that come together to make a clean database build and which determine not merely the structural and data accuracy, but also the performance and security characteristics of a database.

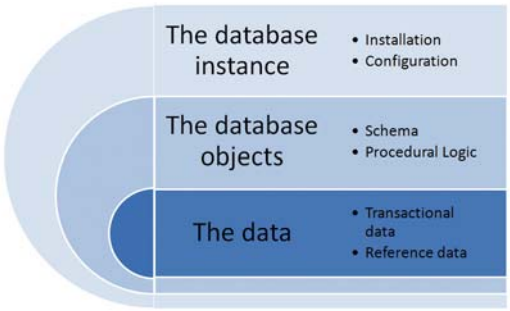

Figure 1

Database design layer

It is important that the business case for modernizing one's strategy for Database Change Management, as well as the strategy itself, include the database design layer in addition to the physical implementation. For the design layer (in the form of a data model, or stored procedure constructs) is not merely a precursor to a physical database implementation, but it is also used for software design, impact analysis, and publishing of metadata to a variety of applications (include ETL, SOA, OLAP, etc.) Therefore it is crucial that the scope of the overall database change management include the database design and not just the database. The business case to improve the status quo rests on several propositions.

The first is efficiency: to diminish the high level of manual effort associated with managing database change.

The next is accuracy: being able to state with confidence what objects, settings and data are in what environment at any given time and how a change to one impacts the others.

The first core requirement of a database change management solution is the ability to capture the objects and associated properties of a database into "collections", each with its corresponding intersection in space and time.

Finally there is audit trail: being able to say what a database looked like yesterday, the day before, or a month ago. All three will result in direct and indirect cost savings through improved efficiency and accuracy. The last two however can also be justified by regulatory compliance.

Now that we have defined the problem space, we can begin to look at some of the characteristics of the solution. The easiest starting point (easy because it tends not to differ so much across organizations… it is the processes themselves which are very organization dependent) is to discuss what possibilities exist out of the box for supporting your database change management program. This should equip you with the tools you need to begin revisiting and improving your processes.

The first core requirement of a database change management solution is the ability to capture the objects and associated properties of a database into "collections", each with its corresponding intersection in space and time. Again, this should include not merely schema objects, but also server and database specific configuration settings. In the screen capture below can be seen a sample collection, which for the sake of example includes just tables and indexes on a Sybase 15 server:

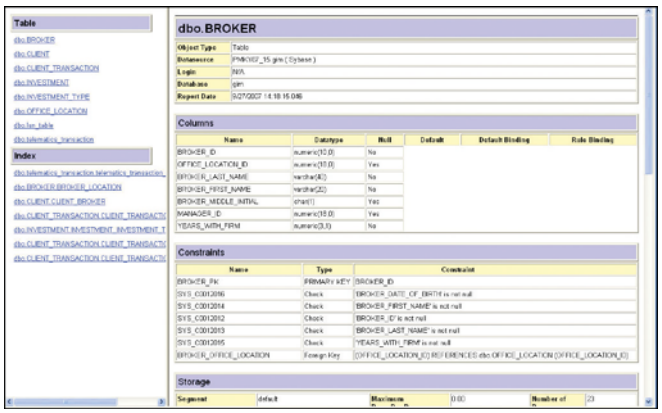

Figure 2

Being able to compare between a collection and a live database is an important requirement, which builds upon the "collection" concept, as is the ability to compare between collections (and also between live databases). In a Sybase replication environment, the ability to compare the multiple databases is the cornerstone to ensuring database integrity. Comparison operations should be flexible however, in order to pinpoint problems, and ensure efficiency. It should be possible to narrow the scope of a particular comparison operation horizontally – by selecting what objects are included in the comparison – and vertically – by selecting what object characteristics are being subject to comparison.

Foundational steps

Identifying differences and generating reports is another foundational step. This can to a certain degree be accomplished with software change management tools, where archived DDL files are "diffed" against one another. The database change management tool takes this two steps further however: first it can reverse engineer what is currently inside of live database, versus merely comparing between archived DDL files. However it also offers the ability to generate an alter script to implement the change, saving a great deal of manual effort when implementing change.

Modern database change management tools should be able to generate reliable, syntactically-accurate and properly-ordered SQL to bring the desired components in line with the comparison target. It should also be able to preserve any existing data and structures, regardless of whether a table must be dropped and recreated and preserve dependent objects, referential integrity, grants, etc., and recompile any dependent objects.

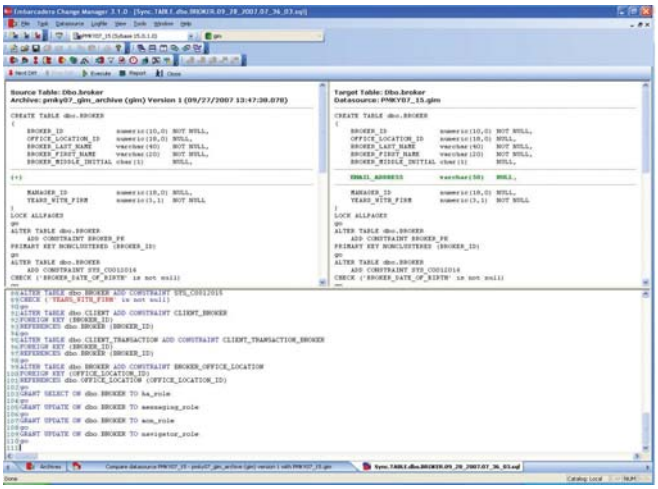

Figure 3

Figure 3 is an example where a column, EMAIL_ADDRESS, has been added to the same Sybase 15 table as above, with the archive DDL to the left, the modified table to the right and an alter script below.

Database settings should be dealt with in a similar fashion, however with settings, it can be useful to have a specialized type of archive (called a "standard") that serves as a baseline or template against which various classifications of systems should be measured.

Below is an example of a comparison between a standard settings template for a development database, and the actual settings for one of the development databases, with differences highlighted:

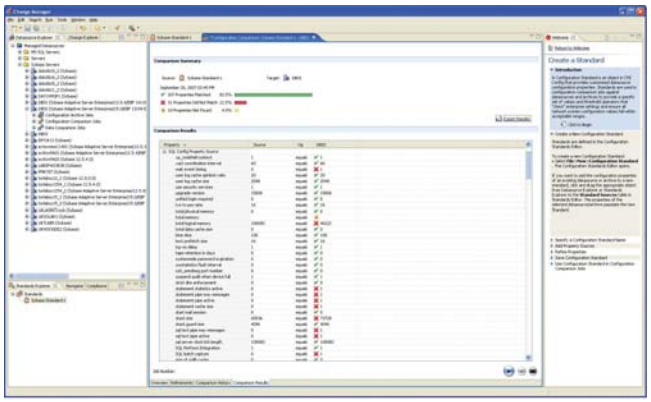

Figure 4

If the management of structural change within a database is likened to maintaining a building's physical structure, the management of data change is that which ensure that all of the fixtures are in the right place so as to meet the needs of the occupants. Data change management is the process by which one may compare and verify data: within the same database, between databases, or even across different database versions or platforms.

Incomparable data compares

In organizations where reference data is used across databases, data compares can prove invaluable. Though the structure of the tax_rate table, for example, may be identical, a difference in data could lead to serious problems. In replication shops, the ability to compare sub-sets of the entire dataset provides a means to validate the replication jobs. As with structural comparisons, the ability to limit the scope of an operation to particular set of tables, as well as a particular set of columns within the tables, is important to productivity and performance.

Below is an example of a data compare, of two databases residing on different servers. Here, the production database (12.5.4) is running in parallel alongside a cut-over database (Sybase 15), until the system is ready to be cut completely over to Sybase 15. The scope of the compare is a single table. The results show that five of the rows did not match:

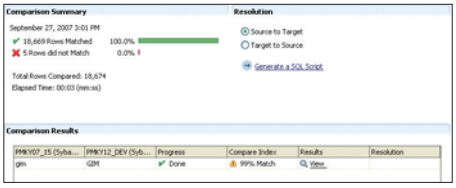

Figure 5

Drilling down, one can see which specific rows did not match, and bring them into sync by selecting the rows of data to be carried over, at which point update, insert, and/or delete statements will be generated, as appropriate:

Figure 6

The technologies described above, available in today's state-of-the-art database change management tools, elevate the game significantly from common homegrown approaches, such as managing DDL within directory trees, using software source control to manage database structures, or considering the database or backup to be the "master copy".

Equipping oneself with a robust set of tools is an important step in developing a modern and effective database change management solution. Also of great importance is an unambiguous change process, which considers the various origination points of change, changes in responsibility for various object types across the project/database lifecycle, standards around tool usage and considers the design layer in addition to the physical implementation. With a clear vision, and judicious use of process and technology, it is possible to craft a robust and systematic solution for managing database change across one's enterprise, yielding significant benefits in a very short time. ■

www.embarcadero.com