

The White Papers

Handling Change in a Complex Multi-Database Environment

By Mark Gurry

Contents

<i>Introduction</i>	3
<i>Background</i>	3
<i>Migrating through the Development Life Cycle</i>	4
<i>Rolling Back Changes Gone Wrong</i>	6
<i>The Need for Auditing</i>	6
<i>Controlling Schema Changes using Formalized Releases</i>	7
<i>The Solution</i>	8
<i>Making Complex Tasks Simpler</i>	8
<i>Auditing</i>	9
<i>Ease of Use</i>	11
<i>Conclusion</i>	11
<i>About the Author</i>	11

Handling Change in a Complex Multi-Database Environment

By Mark Gurry

Introduction

This paper discusses the major issues that must be controlled to successfully manage schemas in a complex multi-database environment.

Background

Prior to the advent of Oracle7, the majority of Oracle sites had databases less than 1 GB and fewer than 200 concurrent users. The DBA usually had enough of a time window overnight to perform cold backups of the database as well as full exports. If schema changes were required, the DBA had a comfortable time window to repair problems if unforeseen problems were to occur. In 1998, large sites are measured in terabytes, not gigabytes. Today, many Oracle sites have thousands of concurrent users. Many of the sites are 24x7x365 sites, with little time for a DBA to perform routine maintenance and application upgrades.

If errors occur during schema upgrades, there is too little time to recover. All hell breaks loose when the end users are told that their application will be off the air for a while until schema problems are corrected. Organizations today are more dependent on the Oracle databases for their mere existence. Added to this, there is a growing expectation to develop and implement applications faster than they have ever been implemented before, placing pressure on the DBA to perform the schema migration tasks quickly and accurately.

Prior to Oracle7, sites had tables, unique and non-unique indexes, sequences, views and that was about it, apart from the grants and synonyms that had to be maintained. The databases did not contain stored code such as packages, procedures, functions and triggers. They occasionally contained constraints, but only as documentation.

With the arrival of Oracle7 and the Client/Server explosion, databases that contain stored code are the norm, not the exception. Packages, procedures, functions and triggers are used heavily. The dramatic functionality and performance improvement in Oracle procedural language, PL/SQL, has made it appealing as a serious development language. The movement of the stored code through the development life cycle (development, test and production) has added an extra dimension to the migration process.

Most sites in 1998 make heavy usage of constraints, including primary and unique key constraints, foreign key constraints and check constraints. Add in the heavy usage of default values on columns, data replication using snapshot logs and snapshots and the DBA has one complex role to perform.

Most sites now make heavy usage of roles. Roles on top of roles are not uncommon. Then you have views of views, views that contain functions, pity the DBA. And to add salt to the wound, managers usually sit back wondering how the DBA is actually spending his/her time. DBA's are often forced to continually justify their mere existence.

Migrating through the Development Life Cycle

New application releases come from many sources. Some applications are bought as packages from third parties. Others are developed in-house using Case tools. Some applications are developed internally using contract staff. Others are developed using full time internal staff. Others, a combination of contract and full time staff. Regardless of the origin of the application, the schemas are the responsibility of the DBA. If you are unsure if this is true, wait for a problem to occur and see who gets contacted.

The task of migrating programs, database schema changes and data through the development life cycle has always been a complex one. It requires careful planning and coordination to perform the task effectively. There are few products that assist the total migration task. The products that do exist are usually unable to assist with the more complex schema changes that are required for an application, such as assisting the DBA to write a script to drop a column from a table.

Management is often convinced by salesmen that products that simply check-in and check-out programs and “manually developed” DDL scripts will save the day. This could not be further from the truth.

If your new schema object scripts are placed into RCS, SCCS, PVCS or another source code control package, it is often difficult to piece together the history of changes on a selected object, such as a table. The products do not provide a facility to view the full change history of a table. Retrieving the historical information is a tedious and time-consuming task.

Manually generated DDL scripts will require different storage and privilege requirements from one environment to the next. It is unlikely that Test will have the same storage requirements as Development, or Production the same storage requirements as Test. The privileges may or may not be the same depending on the usage of roles.

The more complex, time-consuming and potentially error prone schema management tasks that a DBA performs include the following:

1. Deleting a column from a table
2. Renaming columns
3. Changing a data type
4. Shrinking the length of a column
5. Reordering columns
6. Adding privileges to a release that has many tables, views, packages, procedures, functions, sequences
7. Adding storage to a release that has many tables, indexes and primary and unique key constraints
8. Rebuilding a table or other object to eliminate fragmentation or chaining, or to move another tablespace
9. Adding not null columns to tables that already contain data

There is a lack of tools on the market place to assist DBA's with any of these tasks. To understand the complexity of the task at hand, consider Task 1, deleting a column from a table. Until we're blessed by Oracle with this functionality, the task of deleting a column is a complex one that requires the following steps:

1. Rename the table - for example, EMP to EMP_BKUP
2. Drop all foreign key constraints that this table uses
3. Drop all foreign key constraints that reference this table
4. Drop all primary and unique constraints on this table
5. Drop all check constraints on the table
6. Drop indexes off the table
7. Drop all triggers on the table
8. Drop all privileges off the table
9. Create the table without the unwanted column and with the appropriate storage
10. Insert into the new table selecting the required columns from the EMP_BKUP table

If the table contains LONG or LONG RAW columns, you will need to either write a PL/SQL procedure or use the COPY command to insert the data.

11. Reapply the unique and primary constraints
12. Reapply foreign keys that reference this table
13. Reapply the foreign keys on this table that reference other tables
14. Reapply the indexes
15. Reapply the triggers
16. Reapply the privileges
17. Reanalyze the tables, indexes and constraints if required

Interestingly, renaming columns, changing a column's data type, shrinking a column's length or rebuilding existing tables all require each of the 17 steps to be performed. If you are replicating using snapshots, you also have the task of dropping the snapshot log off the backup table and recreating the snapshot log on the new table. It is becoming increasingly obvious that if there are one or more schema changes that fit any of the mentioned criteria, the DBA really has his/her work cut out. One simple column rename may require over 100 lines of DDL code.

There is a growing trend of not deleting columns that should be deleted, not renaming a column that should be re-named, not changing a data type that should be changed, not reducing the length of a column that should be reduced, not reordering columns that should be re-ordered and not rebuilding a table that is badly fragmented or experiencing severe chaining.

One reason for avoiding the task is the sheer complexity and the potential for error, not to mention the excessive amount of time required to perform the task. Another reason is that inexperienced developers and DBA's often handle schema changes. They are often not totally aware of how to code the changes. Some sites have the developers create the DDL scripts. They are then handed to the DBA section for manual review. This task is usually assigned to the most junior DBA because the senior DBA's are working on what seems to be a more important task. The outcome is often a disaster!

If the schema change is not made, you have created potential for confusion. A column may be called one name in one table and the same column may have another name in a different table. Renaming the column is a difficult task, and is often avoided. Developers required to maintain the code will often be confused by columns that should have been renamed.

Columns that should have been deleted from a table can cause even more severe problems. If a developer is new to a project, they may populate the column just in case. Perhaps the developer may even index the unwanted column. The extra column will also consume extra disk storage.

When columns are added to a table at random, with little thought to the logical ordering of columns, developers may be confronted with a horrendous task of trying to decide how columns relate. A column, for example, PROJECT_NO with conflicting data type or length from one table to the next can cause rows not to be returned when the tables are joined using the columns. The problems mentioned are real and common and happening right now at sites that we have health checked.

Adding storage and privileges to objects for either a new release or for a totally new environment is one of the most tedious and potentially error prone tasks that a DBA can perform. After you have sized the first 350 of the 600 objects, the task starts becoming extremely boring. Performing tasks while bored can only increase the potential for error.

Some sites put a large amount of effort into the sizing, documenting each object's storage in a Case tool. The sizing is adequate for production, but is obviously a little large for the development, test and training databases. It also may not suit sites that have multiple production sites of varying sizes. Invariably the DBA will have to spend time, often many hours, manually adjusting the storage to suit each environment.

The method that many sites use to size objects is to categorize tables into small, medium and large. There may be a couple of extra large tables that are sized individually. Tables are often sized using this method in the Development and Test environments.

The Test environment is often used as a base for the creation of the object sizing in production. Table growth is monitored and used to size the production database. The ideal way to size is often to scale the storage “n” times larger than Test - for example, all Production tables may be required to be ten times the size of Test, and Test may be four times the size of Development. Tables experiencing chaining in the Test environment, as identified by the CHAIN_CNT in USER_TABLES view, will have the PCTFREE increased.

Rolling Back Changes Gone Wrong

Applying changes to the production database using overnight schedulers is not a professional practice if you cannot be absolutely certain the changes will proceed without error. Imagine the reaction at Wall Street if the first five of the schema changes worked correctly and the next five failed. I wouldn't like to be the DBA at the Stock Exchange on the morning of the disaster. It is essential that every change made to a schema can be rolled back.

If you consider that rolling back adding a column equates to deleting the column that has been added, the rollback task can require a significant amount of DBA effort to provide the rollback scripts. Most sites do not bother creating rollback scripts because of time constraints. This often proves disastrous as the sites take on more and more critical applications.

The Need for Auditing

With the Oracle RDBMS being used for critical 24x7x365 applications, it is essential that auditing take place to ensure that schema changes will work correctly prior to making the changes. It is just as essential that the database is checked to ensure that the schema changes that are being applied have been applied successfully.

The schema audits that a competent DBA must be able to perform are as follows:

1. Audits to ensure that the schema changes will run correctly prior to the changes occurring. This audit includes:
 - checking the object owners privileges and quotas to allow the creation of the new objects,
 - ensuring that all new objects have been given storage and privileges where required,
 - ensuring that all tablespaces specified have sufficient free space and large enough contiguous free extents,
 - checking that all objects exist as expected (there is no point adding an index to a table that doesn't exist), and
 - checking that all changes are applied in the correct sequence - for example, you can't drop a primary key off a table until all foreign keys that reference the constraint have been dropped or disabled.
2. Audits to ensure that all schema changes that should have been applied have been applied successfully.
3. Audits across database owners to detect differences and ensure that the differences are as expected.

A DBA should be able to quickly list schema differences between the Development and Test database, or between the Test and Production, perhaps even comparing one Production database's objects to another Production database's objects. In a DBA questionnaire, this functionality was recently voted the most sought-after of all new functionalities.
4. Audits on a selected database owner to ensure that the owner has the objects expected
At any given point in time, the DBA should be able to ensure that the objects existing in an environment are as he/she expects. To achieve this check, the objects must be stored elsewhere to provide the comparison. This check will provide information on missing indexes, as well as unwanted objects such as temporary backup tables that should have been removed from the database.
5. Audits on the full change history of all schema objects. The audit information should include when the various object changes were applied to each database.
Ideally the DBA should be able to view tables or other database objects to see the full history of changes:
 - What date was the column dropped?
 - What release was the column dropped in?
 - Which environments has the change been applied to?
 - What application version corresponds to the change?
 - Which other DDL changes need to be applied with the drop column?

All this should be viewable online, not by spending half a day searching for the appropriate hand written DDL which may be buried somewhere in a "check-in – check-out" version control software.

Controlling Schema Changes using Formalized Releases

Now that Oracle has entered the arena of serious, large and mission critical applications, it is not practical or professional to log directly into an environment such as the production user and commence making direct schema changes. Many sites that run mission-critical applications

have usually downsized their hardware and are using Oracle for the first time, or have used Oracle for the less critical databases in the past.

Mission-critical databases tend to be closely audited, with the auditors insisting that each new schema release has been thoroughly checked to ensure that it won't fail. They also insist that each schema change implemented has an associated rollback script.

The common way to perform releases on mission-critical databases is to use formalized release identifiers. Each schema change must have an identifier. One identifier may require many new, modified or deleted tables or any other database objects. Ideally the same identifier will be used for the programs that need to be migrated with the schema database changes. The same identifier may also be used for reference data that needs to be migrated with the programs and schema changes.

Many minor releases may occur to the development environment. The minor releases may be joined, and a larger release that combines many of the smaller releases can then be applied to Test. An even larger number of minor releases may be combined and applied to the production environment. An effective DBA team will be able to assist the person coordinating the release by providing a list of schema differences between the environments being migrated from and to. The list can then be reviewed to ensure that no required schema changes are absent.

The Solution

Schema Manager by Quest Software is not a tool to replace the DBA. It is a tool that will assist the DBA greatly in Schema Management. It automates the tasks that are most time-consuming and potentially error-prone. It greatly reduces the chance of error. Tasks that are difficult to perform, such as deleting columns from tables, are easy to administer. The product can be used equally effectively for small databases as well as the very largest databases.

Mission-critical 24x7x365 databases are supported by the product through extensive pre-audits to ensure that changes can be made successfully prior to making the actual changes. The product provides auditing that fulfills the most stringent auditing requirements. The audits are maintained automatically with no additional effort from the DBA. Every DDL apply script has a rollback script automatically created in case the unforeseen occurs.

The product uses a repository to prepare all changes prior to applying the changes to the real world. The repository provides an extensive audit trail of all changes made to your development, test and production environments. Each change can be pre-audited to ensure that the change can be applied correctly prior to applying the change. After the schema modifications are made to an environment, they may be checked using post-audit scripts to ensure that each schema change has been applied correctly.

Making Complex Tasks Simpler

Some of the functionality that the product provides includes:

- Comparing “real world” database schemas - for example, Development and Test, and creating scripts to make the databases identical.
- Marking a column for deletion and having all of the necessary DDL generated automatically for you with the necessary storage and privileges with a rollback script also created.
- Renaming a column and having all of the necessary DDL generated automatically for you with the necessary storage and privileges with a rollback script also created.

- Changing a column's data type - for example, from a number to a varchar2 - and having all of the necessary DDL generated automatically for you with the necessary storage and privileges with a rollback script also created.
- Shrinking the length of a column - for example, from number(10) to number(6).
- Copying storage from an existing table, index or constraint to new tables, indexes and constraints. Storage can be copied and pasted or dragged and dropped on to many hundreds of objects at once.
- Copying privileges from an existing table, view, sequence, function, package, procedure, or function to new tables, views, sequences, functions, packages, procedures, or functions. Privileges can be copied and pasted or dragged and dropped on to many hundreds of objects at once.
- Create tables like other tables. This is ideal for setting up audit tables. Create copies of hundreds of selected tables at one time. Assign the tables a prefix or suffix of your choice, for example _AUD.
- Export a selected view, package, procedure, function, trigger, sequence, index, or constraint from one environment to another - for example, from the development database to the test database. The Oracle export facility does not allow you to individually select any of the mentioned objects.
- Select one or as many objects as you desire from the real world database and automatically create re-build DDL scripts. This is useful for objects that are badly fragmented, experiencing chaining or need to be re-located to a new tablespace to assist disk loading. The scripts include all required objects such as foreign key constraints that refer to this table.
- Filter all objects for a selected tablespace and rebuild them. You can also filter tables where the table name is like a selected table name, for example, DW%'.
- Scale storage. This enables you to create the objects in Test four times the size of the objects in the Development database. The scaling factor is user-driven. You can scale by tablespace, object type, or where the object name is, like a name, or for a specific object.
- DDL scripts are ordered to apply the largest object first for any new release. This makes better use of free extents in your tablespaces.
- Create a user like an existing user with all necessary privileges and synonyms.
- Create a role like an existing role with all necessary privilege and synonyms.
- Create scripts automatically to have the test object privileges the same as development.
- View which users have particular access rights - for example, which users have DBA access.
- Select an object and view all current privileges on the object.
- See which objects do and don't have privileges on them at a glance.
- Ensure that all DDL steps are performed in the correct sequence - for example, drop the foreign keys that refer to a primary key before dropping a primary or unique key
- Compare environments to identify unwanted objects and create the DDL to drop them.

Auditing

Schema Manager has comprehensive auditing commonly used for mainframe applications such as banking, stock exchange and other mission-critical applications. The audit trails will assist you greatly with ISO-9000 and other standards accreditation. The auditing is maintained automatically with no extra effort on your part. The auditing is extremely useful for any site creating production applications.

Schema Manager uses Delta numbers to control the creation and applying of all database schema changes. Any environment - for example, the Personnel Production environment - can be selected to view the deltas that have been applied to it as well as the actual DDL scripts used. Knowing which deltas have been applied to an environment, and the contents of the delta, enables you to quickly audit the real-world environment to make sure that the objects are as you would expect.

The auditing provided by the product includes the following:

1. Extensive pre-audits prior to applying schema changes to ensure that:
 - The necessary owner privileges and storage quotas are in place to create the objects
 - All mentioned tablespaces exist
 - There is enough total storage in each tablespace
 - The extent sizes are large enough to suit each new object (Schema Manager simulates the way that Oracle applies the objects with free extents being reduced in size as objects are created)
 - Each object that should exist does exist
 - View all indexes on a table at once. This assists with ensuring any index being added is reasonable and does not make an existing index superfluous
 - View code before and after changing it, all on a single screen. The code includes view, functions, triggers, packages and procedures. This functionality is similar to the diff command in Unix.
2. Perform a post-audit to ensure that all changes that should have been made for a delta have been made
3. Audit the real-world to view information that will assist pre- and post-audits, including:
 - View the contents of any schema object in the real world
 - View each tablespace, its free space and largest free extent size
 - List any objects that will crash if they throw an extent
 - List all indexes, primary key constraints and unique key constraints on a table at a glance
 - List all stored objects that rely on this object and all objects that this object relies on - for example, which a view EMP_VW may rely on the EMP table and the EMP_PKG package may rely on the view EMP_VW
 - Which objects are in which tablespace
 - View all of an object's privileges at once
 - View the storage attributes of a selected object, including all storage assignments as well as how fragmented the object is
4. Comprehensive audit all objects stored in the Schema Manager Repository
 - Every change made to any object type is fully audited and viewable online - for example, every column is displayed for each table with the prior and after changes that have occurred. If a column has been renamed, you can see what the column used to be called. If it had its data type changed, you can see what the data type used to be prior to the change.
 - Locks are placed on changed objects to ensure that old versions of the object is not changed by mistake and re-applied to the real world.
5. Audit each physical database environment to check that the schema contents in each database as you would expect. This is made possible through storing all changes in the

Schema Manager repository as well as which changes have been applied to each environment.

Ease of Use

Beta sites are consistently reporting that, on top of being the most powerful product of its type on the marketplace, Schema Manager is also "fun to use." Tasks that normally require hours (and sometimes days) to perform are reduced to a few minutes. This is achieved by features such as being able to copy the characteristics of one object and paste it onto numerous objects all at once. This approach is used for applying storage and privileges that exist on current objects and applying them to many new objects that exist for a new release.

The product uses drag-and-drop techniques to drag down information between the real world and the Schema Manager repository. To make the Development schema objects the same as the Test schema objects, you simply drag down the Test objects into the repository, and then drag down the Development objects into the repository. This automatically creates the differences, which can then be dragged up to the real world and applied. If the differences are objects that need to be removed, such as temporary tables, you can utilize the Schema Manager rollback scripts to remove the objects.

Conclusion

Whether your site is large or small, Quests Schema Manager is a tool that will provide great assistance to Oracle DBA's. It has been designed by DBA's for DBA's. It significantly reduces the amount of time taken to perform the more tedious and error prone tasks. This will free the DBA to perform tasks such as architecture planning and review, investigating new Oracle releases and how the new release features can be used best at your organization, performance tuning and many other valuable tasks. When you start using Schema Manager, you will wonder how you ever did without it.

About the Author

Mark Gurry, a long-time DBA and developer, designed Schema Manager. He is best known for co-authoring *Oracle Performance Tuning* - a bestseller with over 100,000 copies sold worldwide. Gurry is currently a self-employed consultant with Mark Gurry and Associates, a firm specializing in helping organizations develop and administer complex Oracle applications.

To learn more about Quest Software, visit our Web site at www.quest.com.