

This list is based on the articles [“Rethink five outdated ideas about Oracle”](#) and [“Five more things to unlearn about Oracle.”](#) by Bob Watkins, OCP, and includes updated information that covers Oracle 10g.

Database administrators, like most professionals, tend to keep doing those things that have worked in the past. Over time, these practices take on a life of their own, passing down from DBA to DBA like folklore. But all products move on, adding new features, and the Oracle database is no exception. Major new features were introduced in versions 8i, 9i, and 10g that call for a reexamination of what “everybody knows” about Oracle. Let’s take a look at 15 cherished beliefs Oracle DBAs hold and why these beliefs may no longer be true.

Belief #1 Block size is fixed

The fundamental unit of storage in Oracle is the block—the smallest amount of data Oracle can read or write at a time. A block size—2, 4, 8, 16, or 32 KB—is chosen when the database is initially created and is used both for the physical disk space and buffers in memory. Most DBAs believe that once the block size is chosen, it can’t be altered without reloading the database, and that it applies to all tablespaces in the database. Neither is true starting with Oracle 9i.

Each tablespace may now use a different block size. It’s done like this:

```
CREATE TABLESPACE book_data
BLOCKSIZE 8K
DATAFILE '/u3/oradata/prod/bookdata01.dbf' SIZE 100M;
```

You can check which block sizes are in use via the new BLOCK_SIZE column in the DBA_TABLESPACES and V\$DATAFILE views.

If you use a different block size from the default for the database, you must set up a memory cache for it using the initialization parameter file. Six new parameters are defined. db_cache_size replaces db_block_buffers to indicate the size of the buffer cache for default-sized blocks. db_2k_cache_size indicates the buffer cache size for 2-KB blocks, if such blocks are nonstandard for the database. There are 4-KB, 8-KB, 16-KB, and 32-KB versions as well. Unlike their predecessor, db_block_buffers, these parameters are all measured in bytes, not the number of buffers to allocate.

Belief #2 Single extent tables are faster

Oracle DBA folklore says that the best input/output (I/O) performance is achieved when all the blocks for a table are in a single contiguous extent. This one is true—sometimes. But not for the reason you might expect.

In a white paper titled [“How to Stop Defragmenting and Start Living”](#) (free registration required), Bhaskar Himatsingka and Juan Loaiza of Oracle Corporation argue that multiextent tables don’t necessarily perform worse than single-extent ones. What matters, they assert, is the size of the extents. Given a large enough extent size, the Oracle engine can pre-read efficiently even across multiple extents.

To best use this insight, the authors recommend that the DBA standardize on three extent sizes and use them exclusively in all tables: 128 KB for small tables, 4 MB for medium tables, and 128 MB for only the largest tables. Using 128-KB extent sizes may seem to waste space, but the cost of this wasted space is trivial compared to the cost of DBA time fussing over each individual storage allocation. These principles have been built into the new Oracle 8i feature, Locally Managed Tablespaces.

Belief #3 Export and Import are the only ways to reorganize tables

Seasoned DBAs will tell you that when a table needs to be moved, reorganized, or defragmented, it's a painful process. First, the table must be exported to an external "dump" file. The original table must then be dropped, optionally re-created manually in some cases, and finally imported again from the dump file. All this work may be unnecessary as of Oracle 8i.

Using the new MOVE clause in the ALTER TABLE enables you to change the tablespace and/or storage parameters of an Oracle table without using EXPORT and IMPORT. It looks like this:

```
ALTER TABLE author MOVE
TABLESPACE book_data
STORAGE (INITIAL 128K NEXT 128K PCTINCREASE 0);
```

In this case, the TABLESPACE clause tells Oracle to move the author table from its current tablespace to the book_data tablespace. The STORAGE clause works the same as for CREATE TABLE, indicating how large an extent to allocate and its growth properties. Either TABLESPACE or STORAGE is optional, enabling you to move the table without reorganizing it, and vice versa. Caution: In version 8i, the table will be locked for the entire duration of the ALTER TABLE command, so it's still best to do such work during nonpeak hours. In version 9i, that restriction is lifted. When you add the word ONLINE to the syntax, the table can be moved even while users are updating it.

Also, note that you must have enough disk space for two copies of the table: the old one and the new. Oracle doesn't drop the old table until the new one is completely built. If your table is too big for this approach, you'll have to do it the old-fashioned way with Export and Import.

Belief #4 Columns can't be dropped

Oracle DBAs have gotten used to the fact that once a column has been defined for a table, it can't be renamed or dropped. To get rid of it, you have to create a new table without the column and then load that table with data from the original. Finally, you have to reset all permissions, indexes, triggers, etc., on the new table that were on the original one. Not anymore. Starting in Oracle 9iR1, you can drop a column and add the correct one instead. The SET UNUSED, and DROP clauses of the ALTER TABLE command are used for this. Here's an example:

```
ALTER TABLE author
DROP (birthplace, birthyear);
```

```
ALTER TABLE author
SET UNUSED (birthhospital);
```

Both clauses permanently delete a column. The only difference occurs when Oracle does its cleanup. A column that is set to unused disappears from the data dictionary, so users can't reference it. But the physical space is still taken up until manually cleared by the DBA. With the DROP option, the reorganization is done immediately. The SET UNUSED option allows the DBA to make the column unavailable immediately, without inconveniencing users with the overhead of cleanup.

Caution: All data contained in the column is irretrievably lost when a column is marked unused or dropped. These are data definition language (DDL) commands, so there is no ROLLBACK. Be careful and have good backups!

Belief #5 Stored procedures always run as the owner

When an Oracle user is given the permission to execute a stored procedure, he or she is implicitly given permission to do whatever that stored procedure does. No matter how many tables the procedure updates, or how it updates them (even deleting rows), the user can do it. In other words, the user obtains all the rights of the owner while running the procedure. Starting with Oracle 8i, however, a stored procedure can be created with either the rights of the owner or the rights of the person executing it. You use the AUTHID clause of the CREATE PROCEDURE command for this. For example:

```
CREATE PROCEDURE count_authors
  (num_books OUT NUMBER)
  AUTHID CURRENT_USER
IS
  SELECT COUNT(*) INTO num_books
  FROM author;
END;
```

A stored procedure defined as AUTHID CURRENT_USER will allow access to a table only if the user owns the table or has been given permission to use it. Furthermore, references to unqualified table names, like author in the example above, refer to the *user's* copy of author, not the original owner's. The currently logged-in user's schema, or list of objects, is used to resolve references.

Belief #6 Only the DBA can recover data

People who work directly in the SQL language—DBAs and IT consultants—can corrupt or lose data with one mistyped command. In fact, user error is the most common reason for database downtime, according to Oracle. A table dropped from the production database instead of development can bring an application and all of its users to an abrupt halt. Even an improper update can corrupt the results reported from a database. Recovering from such errors used to be a time-consuming job that only the DBA could perform. But since Oracle 9i, users can fix many such errors themselves via SQL commands. The mechanism for this is the new 9i feature called Flashback Query.

Here's an example using the sample data in the [SCOTT schema](#). An employee record is deleted, and the change committed:

```
DELETE FROM emp WHERE empno = 7934;
COMMIT;
```

The row is missing from further SELECT statements, and even a ROLLBACK command cannot bring the row back. However, a Flashback Query can display the contents of the table as it was 10 minutes ago, when the deleted row still existed:

```
SELECT * FROM emp
  AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL '10' MINUTE)
 WHERE empno = 7934;
```

This SELECT statement can be used as the subquery of an INSERT statement to reload the deleted data. Be aware, however, that INSERT will be subject to any constraints on the table, and that any INSERT triggers on the table will be executed. To set the entire session to flash back to a particular point in time, use the DBMS_FLASHBACK package:

```
EXEC DBMS_FLASHBACK.ENABLE_AT_TIME(TIMESTAMP 'yyyy-mm-ddhh:mi:ss');
```

Data accessed during a flashback session cannot be modified, only read. It's just like science fiction stories involving time travel: You can visit the past, but you can't change it! To return the session to the present, type:

```
EXEC DBMS_FLASHBACK.DISABLE;
```

For Flashback Query to work, the database must be using Automated Undo Management (AUM), and an undo tablespace must be created. The amount of time a user can flash back is limited by the initialization parameter undo_retention and the size of the undo tablespace. Although this feature is a godsend for IT consultants, Oracle's intent is for any user who directly types SQL to be able to recover from his or her own errors. Flashback is an object privilege, so it can be granted on individual tables or to all tables via the system privilege

```
FLASHBACK ANY TABLE
```

But wait: it gets better! In version 9i, Flashback is limited to Data Manipulation Language (DML) commands such as SELECT, INSERT, UPDATE, and DELETE. But in Oracle 10g, even a dropped table can be recovered via Flashback.

Belief #7 Oracle can't store fractions of seconds

Oracle's DATE datatype has always stored time to the nearest full second. Developers needing more precise time measurements use the NUMBER datatype instead. This practice makes computing time intervals difficult.

Starting with version 9i, Oracle includes an enhanced date/time datatype consistent with the 1999 SQL standard. To declare such a column, use the TIMESTAMP datatype and indicate the number of fractional digits desired (the default is 6):

```
CREATE TABLE event_ts (  
  event_id NUMBER(6)  
  ,event_name VARCHAR2(40)  
  ,start_time TIMESTAMP(2)  
  ,elapsed_time TIMESTAMP(2)  
);
```

A TIMESTAMP literal, like a DATE literal, must be enclosed in single quotes. Unlike dates, however, the word TIMESTAMP is required as well. The following literal represents March 23, 2004, at a half-second past midnight:

```
TIMESTAMP '2004-03-23 00:00:00.50'
```

Although the standard DATE literal doesn't include time, the standard format for a TIMESTAMP literal requires it. The session parameter NLS_TIMESTAMP_FORMAT controls the format, in the same way that NLS_DATE_FORMAT sets the standard DATE format. A new conversion function, TO_TIMESTAMP, creates a TIMESTAMP from other input formats, and the TO_CHAR function has been enhanced to display a TIMESTAMP's components in any order.

To obtain the current date and time in TIMESTAMP format, use the SYSTIMESTAMP function instead of SYSDATE. For example:

```
SELECT SYSTIMESTAMP FROM DUAL;
```

Belief #8 A corrupted block requires dropping an object

IT consultants dread the Oracle error message ORA-1578, "Oracle data block corrupted." The internal structure of one of the database's blocks is no longer correct. The message identifies the block in error by file number and block number. The cure has always been to run a query such as:

```
SELECT owner, segment_name, segment_type
FROM dba_extents
WHERE file_id = <filenumber>
AND <blocknumber> BETWEEN block_id AND block_id + blocks - 1;
```

where <filenumber> and <blocknumber> were the numbers from the error message. This query indicates which object contains the corrupted block. Then, depending on the object type, recovery is either straightforward (for indexes and temporary segments), messy (for tables), or very messy (for active rollback segments and parts of the data dictionary).

In Oracle 9i Enterprise Edition, however, a new Recovery Manager (RMAN) command, BLOCKRECOVER, can repair the block in place without dropping and re-creating the object involved. After logging into RMAN and connecting to the target database, type:

```
BLOCKRECOVER DATAFILE <filenumber> BLOCK <blocknumber>;
```

A new view, V\$DATABASE_BLOCK_CORRUPTION, gets updated during RMAN backups, and a block must be listed as corrupt for a BLOCKRECOVER to be performed. To recover all blocks that have been marked corrupt, the following RMAN sequence can be used:

```
BACKUP VALIDATE DATABASE;
BLOCKRECOVER CORRUPTION LIST;
```

This approach is efficient if only a few blocks need recovery. For large-scale corruption, it's more efficient to restore a prior image of the datafile and recover the entire datafile, as before. As with any new feature, test it carefully before using it on a production database.

Belief #9 Columns can't be renamed or reorganized

Renaming a table column or changing its data type usually meant creating a new table and copying the old data to it. Columns couldn't be renamed at all, and datatypes could be changed only if they had no data (only NULL values).

Oracle 9i has not one but two ways to overcome these limitations. The ALTER TABLE command can now rename columns directly:

```
ALTER TABLE books RENAME COLUMN ttitle TO title;
```

Function-based indexes and constraints will automatically use the new column name. However, code objects such as views, triggers, procedures, and functions will be invalidated by the change and must be recompiled. Be sure to assess the impact of a renamed column on your code before doing it! Of course, if you have just created a table and there are no dependent objects, this is a quick fix for bad typing.

A supplied PL/SQL package called DBMS_REDEFINITION enables a DBA to change a table's column structure while the table is online and available to users. It's a complex procedure, but in general the steps are as follows:

1. Use DBMS_REDEFINITION.CAN_REDEF_TABLE to check whether the table qualifies for online redefinition, and specify whether the redefinition will be by primary key (recommended) or by row IDs.
2. Create an empty table in the same schema, but with the desired layout. Omit columns you want to drop; include new columns you'd like to create.
3. Use DBMS_REDEFINITION.START_REDEF_TABLE to begin the redefinition process. The parameters to this procedure indicate the old table, the new one, and how to map the existing columns to the columns of the new table.
4. Create any constraints (disabled), triggers, indexes, and grants desired on the new table.
5. Use DBMS_REDEFINITION.FINISH_REDEF_TABLE to complete the process. The original table is locked for a short time regardless of how large or small it is, while the definitions are swapped between the two tables.
6. Drop the temporary table used in the redefinition; it is no longer needed.

Of course, redefining a table doesn't automatically update any application code that accesses that table. Applications must be changed and tested separately. What DBMS_REDEFINITION does, however, is shorten the time that the table is unavailable to users at cutover time.

Belief #10 Only the owner of a table can grant permission to use it

When I explained Oracle security in prior versions, clients couldn't believe that the DBA could not grant permissions on a table unless the table's owner had first granted it to the DBA. Historically, however, this has been the case. The restriction was part of Oracle's design, but it made administration difficult. In Oracle 9i, a new system privilege changes this.

The DBA role now has a system privilege called GRANT ANY OBJECT PRIVILEGE. In the past, a statement like GRANT SELECT ON scott.emp TO giselle;

would fail unless SCOTT had first granted the DBA the SELECT privilege on his table WITH GRANT OPTION. Now, that same statement will work. This privilege can also be used by lead developers to grant permission on a schema's objects without having to log in as that schema's owner.

Belief #11 The only wildcards in SQL are % and _

When doing pattern match queries with the LIKE operator, both DBAs and developers have learned to put up with having only two wildcard characters at their disposal: percent, which matches anything, and underscore, which matches any one character. For more elaborate matching, they would write PL/SQL.

Starting with Oracle 10g, this extra coding is no longer needed. A full set of regular expression syntax, such as used in UNIX shell scripting, is now available directly in SQL. Oracle supports the complete set of POSIX standard extended regular expressions (ERE). For a list of these expressions, see Appendix C of the Oracle 10g SQL Reference.

The operator REGEXP_LIKE replaces LIKE, and the regular expression must be quoted and in parentheses. For example, the following WHERE clause searches for A1, A2, or A3 occurring only at the beginning of a part number:

```
WHERE REGEXP_LIKE (partno, '^A[123]')
```

Notice that no * wildcard is needed at the end of the regular expression. Unlike the wildcards used with the LIKE operator, regular expressions assume a partial match unless you force a complete one. So the above expression will match "A1" and "A3", but also "A234". To force an exact match, use the \$ to indicate end of line:

```
WHERE REGEXP_LIKE(partno, '^A[123]$')
```

The REGEXP_INSTR, REGEXP_SUBSTR, and REGEXP_REPLACE functions extend the INSTR, SUBSTR, and REPLACE functions to use regular expressions in the match argument. Again, the Oracle 10g SQL Reference describes these.

Belief #12 You have to rebuild a table to reset its High Water Mark

End users often wonder why a table with few rows in it can take a long time to search. DBAs know that if the table used to have a lot of rows at one time, the search may be slow because Oracle has to look at every block that used to contain data—up to the table's High Water Mark, or HWM. They may also believe that the only way to reset the HWM is to rebuild the table, either via export/drop/import or ALTER TABLE MOVE.

In 10g, this is no longer necessary. A new feature called Online Segment Shrink can reclaim space in the table and adjust the HWM down as well. The syntax is:

```
ALTER TABLE tablename  
SHRINK SPACE [COMPACT] [CASCADE];
```

Giving this command without the options defragments the table, compacting the rows. It then adjusts the HWM to the new high position and releases the freed-up space.

The COMPACT option does the defragmentation but does not adjust the HWM nor release the storage space. The CASCADE option shrinks not only the named table, but any dependent objects, such as indexes.

And now, the fine print. The tablespace in which the table is stored must be set up for Automatic Segment Space Management, and the table itself must have row movement enabled. Because the moved rows will have new ROWIDs, you should turn off any triggers that fire based upon the ROWID, or they'll be re-executed. There are other restrictions as well: Consult the documentation.

Belief #13 Incomplete recoveries require restoring old data files

Oracle DBAs know that Oracle recovers itself completely from instance failure upon startup and from physical failures, such as media failures, via the RECOVER command in RMAN or SQL*Plus. When a logical corruption occurs, however, they believe the only recourse is to restore the database files from backup taken before the problem occurred and to “roll forward” to the desired time via the redo logs.

In Oracle 10g, another option is possible: to roll *back* the database to a point in time prior to the damage, using the current data files. This feature can save a lot of time in an incomplete recovery scenario.

The Flashback feature introduced in Oracle 9i has been dramatically extended in 10g with FLASHBACK DATABASE (available as a command in RMAN, and a statement in SQLPlus.) With the proper setup, you can now recover the database by rolling it backward from its current state, instead of rolling forward from an older image. This can be much faster, as it works with the existing data files. No old versions need to be restored.

Flashing back the entire database is still an incomplete recovery: You will lose any data entered after the point in time to which you flash back. In brief, here’s how to set it up:

1. Set aside a flash recovery area on disk large enough to hold the flashback database logs and other RMAN backups, such as control files. Set the parameters DB_RECOVERY_FILE_DEST and DB_RECOVERY_FILE_SIZE to tell the instance where to find it.
2. Set the DB_FLASHBACK_RETENTION_TARGET parameter to the number of minutes maximum that you want to be able to flash back.
3. Enable the flashback feature with the database in mount mode, with the command ALTER DATABASE FLASHBACK ON. Then ALTER the database open. The database will automatically begin backing up changed blocks on a regular basis to the flashback recovery area. Think of it as a continuous incremental backup at the block level.

If you need to flash back the database to an earlier time:

1. Place the instance in mount mode
2. Connect to the instance in RMAN and use the FLASHBACK DATABASE command. This command locates the most current block images before the flashback time you request and restores them. Then, it uses the redo logs to roll forward to the exact flashback time. Because the blocks are backed up fairly frequently, there is much less work to do to bring these blocks current. Plus, you avoid the entire time of restoring data files.

This technique is not appropriate for every instance, but like all insurance policies, you pay a little overhead on a regular basis to avoid a much bigger payout in case a problem occurs. For further information, consult the Oracle Database Backup and Recovery Advanced User’s Guide, Chapter 9 (“Flashback Technology: Recovering from Logical Corruptions”).

Belief #14 Tablespaces can be transported only to the same platform

The transportable tablespace feature introduced in Oracle 8i enables datafiles to be copied directly from instance to instance. Because various operating systems store data in different byte orders (“endianness”), many DBAs believe you cannot transport tablespaces to an instance with a different block size or to a different hardware platform.

In Oracle 9i, the block size issue went away because you can have multiple block sizes in an instance. In Oracle 10g, the endianness problem also went away, because you can use RMAN to convert the endianness of the data. The result is a datafile copy targeted for a specific operating system. When such files are transported, they are already in the correct format necessary to plug into another instance.

The RMAN command CONVERT is used for this. For example:

```
CONVERT TABLESPACE example TO PLATFORM 'HP-UX (64-bit)';
```

The view V\$TRANSPORTABLE_PLATFORMS contains information about which are compatible and which require use of the CONVERT command.

The Database Administrator’s Guide, Chapter 8 (“Managing Tablespaces”) gives more details on transporting tablespaces.

Belief #15 **CONNECT, RESOURCE, and DBA are a convenient way to set up users**

Many DBAs still use the CONNECT, RESOURCE, and DBA roles to set up new user accounts, either in automated scripts or manually, just by habit. Sometimes the things we have to unlearn are simple, comfortable things, like an old sweatshirt or pair of shoes that have long since worn out. These legacy roles were introduced in Oracle 7—yes, that's three major versions ago—as a bridge between the simple security model of Oracle 6 and the more granular one we have had since then. But that's all they were: a temporary convenience.

In computer security, the principle of least privilege says that users should have only the minimum privileges necessary to do their jobs. The CONNECT role, for example, includes system permissions such as CREATE TABLE and CREATE SEQUENCE, things that most end users are unlikely to need. The RESOURCE role contains the powerful UNLIMITED TABLESPACE privilege that overrides the tablespace quota system.

A better practice is to analyze the requirements of various job roles and create custom roles to match. Grant the required system and object privilege to these roles and then grant the roles to users. Grant users quotas on tablespaces only if they will be creating objects in those tablespaces. (Although quotas must be set directly on users and not roles, you can simplify the process in Enterprise Manager by using the "Create like" command to clone an existing user account.)

The bottom line

Our skills with Oracle are like the stocks in an investment portfolio. While most of us understand that we have to add new skills on a regular basis, it's not always obvious that there are some we should dump as well. Manage your portfolio well, and you'll have the most efficient set of tools to do the job.

Bob Watkins (MCSE, MCT, OCP, MCDBA) is a computer generalist whose 25 years of experience includes seven years as a full-time technical trainer and another eight years as a consultant and DBA. He is a senior instructor at SQLSoft+, a computer training firm in Bellevue, WA. A "free agent in training," Bob maintains a Web site listing tools and resources for full-time salaried employees who want to go independent without losing their spouses, minds, or shirts.

Additional resources

- Sign up for our [Downloads Weekly Update](#), delivered on Tuesdays.
- Sign up for our [Oracle newsletter](#), delivered on Wednesdays.
- Check out all of [TechRepublic's newsletter offerings](#).
- ["Simulate variable arguments in PL/SQL"](#) (TechRepublic)
- ["Using SQL*Loader to load objects"](#) (TechRepublic)
- ["Oracle's latest software focuses on enterprise application integration"](#) (TechRepublic)

Version history

Version: 1.0

Published: June 17, 2005

Tell us what you think

TechRepublic downloads are designed to help you get your job done as painlessly and effectively as possible. Because we're continually looking for ways to improve the usefulness of these tools, we need your feedback. Please take a minute to [drop us a line](#) and tell us how well this download worked for you and offer your suggestions for improvement.

Thanks!

—The TechRepublic Downloads Team