Iggy Fernandez
NoSQL Explained

Daniel Fink
Real Life Recovery

George LaFramboise
Creating Oracle 12c Databases

Kellyn Pot’Vin
Social Media

Board Focus - Tim Gorman
2014 Scholarship Recipient
Focused on Delivering the right Solution for the right Project

Arisant focuses on understanding business requirements to ensure the most effective implementation of the right solution. Our strategy is aimed at providing an honest and expert brand of consulting services for both the Private and Public Sectors. Contact us today to discuss any of the following services, and more.

- Consulting Services
  - Architecture
  - Analysis
  - Design
  - Implementation
  - Project Management
- Staffing/Support
- Managed Services
- Managed Hosting Services
- Oracle Software Sales
- Oracle Hardware Sales

---

Oracle Experts, Easy to Work With

Oracle Database | Data Strategy | Managed Services
Oracle E-Business Suite | Oracle Exadata | Oracle License Reseller

www.dbak.com | 720-475-8600

Proud to be a Gold Sponsor for RMOUG 2014 Training Days
On the Cover:

Steve Hockett captured this early morning “moon set” on the lower elevation trail approaching the climb up Mt. Elbert. As an avid hiker, and general enthusiast of the outdoors, Steve enjoys the year round challenges and rewards of exploring the vast trails that Colorado has to offer. Mt. Elbert, located in the Sawatch Range, is recognized as the tallest of the Colorado 14ers rising to 14,433 ft. As the moon set, the sun rose providing a beautiful fall day for the climb. “Fall is my favorite season in Colorado because it means ski season is right around the corner”.

Originally from St. Louis (go Cardinals!), Steve has been in Colorado for 15 years and has been part of a great Oracle DBA team at the City and County of Denver for the past 13 years.
Welcome to this edition to SQL>Update!

We have had a very good year through September 2014. Three very successful big events have taken place with one more in the planning stages.

First, Training Days in February was hugely successful. Many thanks to all that volunteered, sponsored, and attended. Planning and preparation for the 2015 Training Days is well underway. This year Kellyn Pot’Vin, (TrainingDaysDir@rmoug.org) is leading and directing the 3 day event.

John Jeunnette, (BoardMemberEmeritus2@rmoug.org) is providing software to help make Training Days another success.

New this year on the Training Days Leadership team, please welcome Karen Funston as the Assistant Director of Training Days. Karen brings a wealth of experience from other Higher Education Conferences to RMOUG.

The volunteers, sponsors and members attending the Training Days make it one of the most valuable conferences in the world of Oracle. We look forward to seeing you there in February 17-19th 2015.

Our spring QEW (Quarterly Education Workshop) was also successful and a new Board of Directors was selected at the Broomfield Oracle site. A very big thanks to our members who have agreed to serve this year. In June 2014 we had a changing of the guard to the RMOUG Board. Your new Board:

The Executive Board

John Peterson, President, I have served on the board for many years and have held several board positions, including Vice-President, Treasurer, Secretary and other volunteer boards positions. I am looking forward to a great year and a tremendous Training Days in 2015.

Vince Giasolli, Vice President/ Director of Newsletters, is taking on two roles this years, first as Vice President and as Director of Newsletters.

Chris Ostrowski, Secretary, has served RMOUG in many positions in the past.

Ron Bich, Treasurer, is returning to the Executive Board as our Treasurer, Ron is also a past President of RMOUG.

RMOUG Board Members:

Andreas Katsaris, Director of Marketing, is new to the Board this year, the new position of Director of Marketing has been created. Please welcome Andreas to the RMOUG team.

Bobby Curtis, Director of Social Media, is returning to the Board and handling Important communication to RMOUG members.

Art Marshall, Director of Information Services/Web Master is returning to the Board and is keeping an incredible eye on the RMOUG website.

Bob Mason, Director of Education/Scholarship, is returning to the board. Leading the efforts to identify and vet scholarship awards.

Chris Chase, Director of Special Interest Groups/Meetup, is continuing his support of special Interests of RMOUG, such as Hyperion and has joined the board.

Members at Large, supporting our user group:

Sruthi Kumar Annamnidu, returning to the board.

Mark James, Member At Large, is continuing his support of the board after serving on the board last year.

Al Gonzales, Member At Large, is adding his support to the board.

Past Board Members and Supporters of this year’s Board: Please honor our distinguished members!

Tim Gorman, Past President
Kathy Robb, Board Member Emeritus
John Jeunnette, Board Member Emeritus
Peggy King, Board Member Emeritus, Past President
Kellyn Pot’Vin, Board Member Emeritus/Training Days Director(Non-voting)
Pat Van Buskirk, Board Member Emeritus
Ann Horton, Oracle Liaison
Heidi Kuhn, RMOUG Executive Director.

“If I have seen further than others, it is by standing upon the shoulders of giants.” - Isaac Newton

We are the stewards of RMOUG for this year. Full descriptions are posted on the WWW. RMOUG.ORG website, under current directors.

We also just experienced a tremendous Summer Event at Elitch Gardens.
For the third year in a row, RMOUG has held a QEW meeting in the morning, providing members excellent Oracle Presentations, breakfast, lunch, libations, and the run of the park, for themselves and members of their families. Planning is already underway for 2015 Summer QEW at Elitch Gardens.

Not to see Elitches is Not to see Denver.

Special Thanks to all of our sponsors.

- Arisant, Business Solutions through Oracle Technology
- Oracle
- Dell Software
- AdvizeX - a Rolta Company
- Delphix

This fall RMOUG will hold the last QEW of the year, we are looking forward to welcoming another great event to the south side at Oracle offices in the Denver Tech Center.

Thank you for your interest in RMOUG, we look forward to seeing you all at the upcoming events.

John Peterson
President RMOUG
September 2014.

Stan Yellott Scholarship Fund

RMOUG Scholarship Mission

To provide educational opportunities to members of the organization about the information technology industry in general, and in particular the technology of Oracle Corporation to include databases, storage, networking and application development, specifically the products and services of the Oracle Corporation.

To collect, post and distribute information about Oracle technologies and other related technologies to members.

To provide members with the ability to help their peers maximize their knowledge and skills working with products in information technology and in particular Oracle products.

To provide a consolidated channel of communication, conveying needs, concerns, and suggestions, for members of the organization, to Oracle Corporation and other vendor corporations involved with Oracle related technology.

To encourage members to present their information technology experiences using Oracle and other products and services.

To provide a consolidated channel of communication between members of the RMOUG and other communities in related information technology industries.

To promote educational opportunities for students of information technology through directed funding and services for educational purposes.

RMOUG is committed to supporting others in the pursuit of technical knowledge.

The Scholarship Fund started in 2001 to encourage future IT professional in their efforts to broaden their knowledge. In 2007, RMOUG voted to rename the scholarship fund to honor the memory of Stan Yellott. Stan was a long time member of RMOUG where he supported the user community by serving on the RMOUG board. Stan focused on expanding Oracle educational opportunities. Stan’s vision was to include high school and college students as the next generation of IT professionals.

For Details, Visit the RMOUG Website
www.rmoug.org

For Details, Visit the RMOUG Website
www.rmoug.org
Creating Oracle 12c Databases

Jump Start

by George Laframboise
LightWorx Technology Consulting

Abstract

This article includes a description of how to build an Oracle 12c Container Database using SQL*Plus and then how to create pluggable databases within a newly created container database. Creating Oracle 12c databases with SQL*Plus is not the only method available, you may also use the Database Creation Assistant (DBCA) or Oracle Enterprise Manager (OEM) 12c Cloud Control. I’ve chosen to create Oracle 12c databases with SQL*Plus to illustrate the concepts involved in creating databases in an Oracle 12c environment. I also find SQL*Plus a great deal more flexible and easy to use than DBCA or OEM 12c Cloud Control.

Introduction

The Oracle Database has undergone a significant evolution to a Multitenant Architecture with version 12c. Oracle 12c container databases allow up to 252 pluggable databases within a container database. There is also a seed database (CDB$Seed) within a container database that may be used to create other pluggable databases. That does not mean that all Oracle 12c databases are required to be container databases. You can still create non-container databases; however, there are significant advantages associated with multi-tenant architecture. There are also uses for non-container databases in a development or test environment. A non-container database may be converted into a pluggable database and loaded into a container database at any time.

There are three primary tools used to create Oracle 12c databases, SQL*Plus, DBCA and Oracle Enterprise Manager 12c Cloud Control. This article discusses creating a container database with SQL*Plus and then discusses how to create a pluggable databases from several different sources.

Creating a Container Database with SQL*Plus

In order to create a container database, start out by deciding on a name for your database. The database name must be 8 bytes or less, ascii characters only, and not conflict with the name of any other database in your environment unless you plan on intentionally overwriting an existing database. Once you do that, set the ORACLE_SID and the ORACLE_HOME of the new database in your environment, then make sure your PATH includes $ORACLE_HOME/bin and the LD_LIBRARY_PATH includes $ORACLE_HOME/lib.

The next step is to configure an init.ora for the new container database. I use an init.ora while creating databases since I can easily edit it to resolve any errors during the creation process. Oracle recommends that you convert to a spfile after creation since it is more flexible than the init.ora. The advantage of a spfile is that system parameters can modified with "alter system" and have their values maintained when the instance is bounced without having to remember to edit the init.ora.

On the other hand, using an init.ora requires editing the init.ora file and bouncing the instance to change a system parameter. Obviously, if you wish to just test a parameter, an init.ora is far less convenient especially in a production environment. Once you create an init<sid>.ora, place it in the $ORACLE_HOME/dbs directory (SORACLE_HOME/database on Windows) so that the system doesn’t have to look for it in another directory.

Initialization Parameters

There are at least 368 possible parameters that can be listed in an init.ora (see V$SPPARAMETER in the data dictionary for the complete list). Most databases will not need more than the 27 basic initialization parameters. The only required parameter for an initialization file is "db_name". All of the other parameters have defaults although some of the defaults are not very suitable for most environments.

"Enable_pluggable_database" is an optional parameter however if you are attempting to make a container database it is required. If you do not set enable_pluggable_database to true in the init.ora then attempt to create a container database in the create script you will get an Oracle error (ORA-65093: container database not set up correctly).

It is not required to put every parameter available in the init.ora you are using. Initialization parameters can be changed or added at a later time when the instance is being tuned. However, db_name, db_block_size and enable_pluggable_database cannot be modified after you have created the database so, at a minimum, the init.ora should contain both of those parameters as well as "enable_pluggable_database=true" if you are creating a container database.

1 Base installs allow one pluggable database, additional pluggable databases require additional licensing fees.
2 The Database Configuration Assistant may also be used to configure existing database options, delete databases, manage templates used to create both seed and non-seed databases and to manage (i.e. create, unplug, delete and configure ) pluggable databases.
3 It is a much better practice to drop the old database first and remove it’s files. Back up any database before it is dropped if there is any chance it might have to be recovered later.
4 You can and should convert the init.ora into an spfile.ora later with the "create spfile from pfile" statement before you are complete and ready to release the new database for use.
5 The change can take affect right away and be non-persistent (scope=memory) or take effect persistently when the instance is bounced (scope=spfile) or take effect right away and be persistent (scope=both). Note that not every parameter can be changed with alter system.
6 See the Oracle 12c Database Reference for more complete information on the parameters, the parameter type, syntax, default value etc.
Listing 1 contains a basic init.ora that can be used to create a container database. I generally add a few other parameters because I think the defaults are inadequate. I want to gather them all in one place and make the parameters perfectly obvious so that others working on the database don’t have to expend much energy looking them up.

A db_block_size of 8kb (8192 bytes) is usually optimal for most systems. Block size cannot be changed later (or db_name or enable_pluggable_database). It is important to give adequate thought to what the block size, db name and enabling pluggable databases should set to.

Memory_target specifies the system wide usable memory for Oracle. Setting memory_target larger than 0 is the simplest way to allow the database to automatically manage and tune memory since automatic memory management will be set if you set a memory_target greater than 0. The database reduces or enlarges the SGA and PGA as needed when automatic memory management is set. The default is 0. When it is set to 0, SGA auto tuning is disabled for deferred auto mode auto tuning requests and allowed for immediate mode autotuning requests. If you don’t set memory_max_target, it is set to the value of memory_target.

Processes are operating system dependent and specify the number of OS user processes that can connect to Oracle. Processes also include the background processes that are running. The value of this parameter must be a minimum of one for each background process plus one for each user process. I generally bump up the number of processes to 250 or so. It can always be modified later but I find 250 a good starting point on most operating systems when I am creating a database.

Open_Cursors specifies the maximum number of handles to private SQL areas that each database session can have at once. I usually bump this to 500 as a starting point. Assuming that a session does not open all of the cursors available to it, there is no added overhead in specifying this value higher. Open_cursors is only the maximum number of cursors that can be opened. Increasing open_cursors does not allocate memory for cursors until the cursors are actually opened. Don’t set this too high though. It is designed to prevent a run-away process from allocating excessive amounts of memory.

I prefer to explicitly specify undo management as auto. Auto is the default in Oracle 12c and has been for several versions. I also like to specify the undo tablespace in the init.ora. I do this because I want to make certain that, if I forget to specify an undo tablespace in my SQL statement to create the database, I don’t end up with the system tablespace becoming a fragmented mess because it is where all the undo data is stored.

Creating a SQL Script to Create the Container Database

The most important thing about creating a container database is “enabling pluggable databases”. If you don’t do that in the create script then the database created will never be a container database. It can be imported into another container database as a pluggable database if you wish but it will never be able to house other pluggable databases without the “enable pluggable database” clause in the init.ora and the create script.

The “seed_file_name_convert” clause can be used to specify the names and locations of the seed’s files (CDB$Seed) as well as System and SysAux tablespaces options (tablespace size, autoextend on etc.) for pluggable databases created from the seed.

Once you place the init<sid>.ora in the $ORACLE_HOME/dbs directory, creating a SQL script to create the container database is next. Start out by explicitly declaring the sys and system users and their passwords. Do that even though the default is to create those users automatically. The same old default passwords that every black hat hacker in the world knows by now are used if you create sys and system by default.

Control files cannot be specified in the create database statement. They are specified in the init.ora however if this is the second attempt at creating a database the “controlfiles reuse” clause may be used to overwrite the control files from a previous attempt. For maxdatafiles I specify 1024. The default is 32 and since there is no real disadvantage to specifying it higher and there is a distinct disadvantage to specifying it lower, I set it to 1024.

Oracle strongly recommends that Unicode character sets be used. Unicode character sets encompass most of the major scripts of the world. Choosing a Unicode character set will likely save both time and money if there is a requirement for multi-lingual data at a later time.

In addition, for Oracle 12c, all pluggable databases in a container database must have both the same character set (char data types) and the same national character set (nchar data types). Any database that you wish to plug into a container database must also have the same character set and national character set. For that reason I choose the AL32UTF8 Character Set. I let the National Character Set default to AL16UTF16. With both a Unicode character set and a Unicode national character set any database that will be plugged into the container database can be converted to Unicode character sets without losing data.

At least five tablespaces are needed for a container database: a system tablespace, a sysaux tablespace (non-essential database metadata, see the v$sysaux_occupants view), an undo tablespace, a temporary tablespace and a users tablespace. Even though it is very bad practice to create tables for an application in the

---

7 It is good practice to backup the spfile into a current init.ora on a regular basis for reference or to be used to recreate the spfile if it is lost.
8 8k is actually the default but this is a good parameter to make obvious. It should be a multiple of the physical block size at the device level.
9 Well, not easily anyway.
10 This is also recommended by Oracle in the Oracle 12c Performance Tuning Guide.

Listing 1 – Sample Init.ora to create a container database.

db_name='MYCDB'
db_block_size=8192
control_files=('/home/oracle/oradata/MYCDB/control1.ctl, /home/oracle/oradata/MYCDB/control2.ctl, /home/oracle/oradata/MYCDB/control3.ctl')
memory_target=1G
open_cursors=500
undo_management='auto'
undo_tablespace='MYCDB_undo'
enable_pluggable_database=true

11 Run the following query from the root container to determine whether a database is a container database: “select name, cdb, con_id from v$database;”. “YES” will be returned by the CDB column if it is a container database.
12 $ORACLE_HOME/database on MS Windows.
13 It can be changed by backing up the control files to trace, editing them and creating new control files for the database but do you really want to go through all that?
14 National Character set is an alternative character set that enables you to store Unicode character data in a database that does not have a Unicode character set using nchar data types.
container database, it can be done. There are a number of $OJDS$ tables\textsuperscript{15} that will be created by the create database statement in the CDB users tablespace so a users tablespace is needed. An index tablespace is unneeded.

**System Tablespace**

Specify “extent management local” on the **system tablespace** to be sure that the system tablespace is locally managed. By default, the system tablespace is created as dictionary managed\textsuperscript{16}. With a locally managed tablespace, contention on the data dictionary is reduced, performance is better because recursive SQL against the data dictionary is not necessary, there is no need to periodically coalesce free space in the tablespace and changes to the extent bitmaps don’t generate undo information. I tend to specify the system tablespace for a container database a little bigger than most at 750 MB.

**Sysaux Tablespace**

The **sysaux tablespace** is an auxiliary tablespace to system and is the default tablespace for many Oracle products and features. I also size this tablespace slightly larger at 750 MB.\textsuperscript{17}

**Mirrored Logfiles**

Keeping in mind that the PDBs share the log files with the container database, starting out with 3 mirrored log files at 100 MB is a good starting point as long as log files are tuned when additional PDBs are added\textsuperscript{18}. Redo log files should be mirrored on separate devices to avoid recovery problems.

**Temp Tablespace**

An **undo tablespace** should be created for every Oracle database. If an undo tablespace is not online, the system tablespace is used for undo management which causes fragmentation of the system tablespace and results in performance degradation. In the case of a container database, the undo tablespace will be used for the container database as well as any pluggable database created within the CDB because PDBs do not have their own undo tablespace. In order to avoid undo management occurring in the system tablespace, an undo tablespace should be explicitly created in the create db SQL.

Undo management was specified as “AUTO” in the init.ora even though it is the default. Considering that the undo tablespace is shared by the container database and the pluggable databases within it, the undo tablespace is created with AUTOEXTEND on and MAXSIZE 1G.\textsuperscript{19}

It would be very surprising if this is the optimum UNDO configuration for this CDB but it will do for now. The statistics in the **V$UNDOSTAT** view as well as the Undo Management Advisor can be used later once pluggable databases have been created and applications have been loaded within them to tune the UNDO for the CDB. Other tuning of the database will be required once there is a better idea of what applications will be loaded into the pluggable databases.

**Undo Tablespace**

An **undo tablespace** should be created for every Oracle database. If an undo tablespace is not online, the system tablespace is used for undo management which causes fragmentation of the system tablespace and results in performance degradation. In the case of a container database, the undo tablespace will be used for the container database as well as any pluggable database created within the CDB because PDBs do not have their own undo tablespace. In order to avoid undo management occurring in the system tablespace, an undo tablespace should be explicitly created in the create db SQL.

Undo management was specified as “AUTO” in the init.ora even though it is the default. Considering that the undo tablespace is shared by the container database and the pluggable databases within it, the undo tablespace is created with AUTOEXTEND on and MAXSIZE 1G.\textsuperscript{19}

It would be very surprising if this is the optimum UNDO configuration for this CDB but it will do for now. The statistics in the **V$UNDOSTAT** view as well as the Undo Management Advisor can be used later once pluggable databases have been created and applications have been loaded within them to tune the UNDO for the CDB. Other tuning of the database will be required once there is a better idea of what applications will be loaded into the pluggable databases.

**Temp Tablespace**

Since this database is being created with a locally managed system tablespace, a default temporary tablespace is created as locally managed with uniform extent size of 1M (the default). Locally managed system tablespaces cannot be used as temporary tablespaces so this tablespace is required. Creating it as the default also has the advantage that every user created will be automatically assigned this temporary tablespace even if a temporary tablespace is not explicitly assigned to them when the user account is created. Remember that all PDBs may share the CDB temporary tablespace although each PDB may have it’s own temporary tablespace.

**Default Tablespace**

It is worth designating a default tablespace in the create database statement so that if users are created without an explicit default tablespace, the system tablespace will not be their default tablespace.

---

\textsuperscript{15} The $OJDS$... tables are data dictionary tables that are related to Very Large Databases and Partitioning. These may move to the syasa or system tablespace in future versions.

\textsuperscript{16} The $DBMS\_SPACE\_ADMIN\_TABLESPACE\_MIGRATE\_TO\_LOCAL("SYSTEM")$ can be used to migrate a dictionary managed tablespace to locally managed.

\textsuperscript{17} The $ORACLE\_HOME/dbs/admin/utlgetxsz.sql$ script may be used to resize syasa, to a more appropriate size later if need be.

\textsuperscript{18} The optimal $LOGFILE\_SIZE$ column in the $vinstance\_recovery$ view contains the minimum recommended size of log files as well as other data that may be used to tune the log files.

\textsuperscript{19} I usually prefer not to use autoextend however since long running queries may need extra undo I specify it for the undo tablespace but cap it with a max size. Considering that this is a container database and all the PDBs share it you will need to resize it as PDBs are added. Keep an eye on this tablespace’s size requirements. If you start getting frequent “Snapshot Too Old” errors, the size of the undo tablespace is the likely problem.
$ perl $ORACLE_HOME/rdbms/admin/catproc.pl
20  -n 1 -l
21  /home/oracle -b catproc $ORACLE_HOME/rdbms/admin/
catproc.sql;

It isn’t required although it is recommended to build the product
user_profile table. Again catcon.pl should be used. The script
builds tables in the system schema that allow DBAs to restrict
SQL*Plus users from using specific commands in SQL. For exam-
ple, there might be some users who shouldn’t be deleting data and
this utility serves as a backup to grants on tables. The command to
run it with catcon.pl22 follows:

$ perl $ORACLE_HOME/rdbms/admin/catcon.pl
22  -n 1 -l
23  /home/oracle -b pupbld -u SYSTEM
24  $ORACLE_HOME/sqlplus/admin/pupbld.sql;

Next on the list is to build an spfile to replace the init.ora file.
This is accomplished with “create spfile <spfile_name> from pfile
<spfile_name>”.

And last of all: Backup the database you just created!

Creating the Container Database

Now that a create script has been constructed for the container
database, put the create script in your favorite working directory
and set the environment20. Check through both the init.ora and the
create database script and make sure that any directory referred
to in either script actually exists. Oracle does not make any direc-
tories during the execution of the create database command and
database creation will fail if any directories referred to in the scripts
are missing.

At that point connect to SQL*Plus as SYSDBA and “startup
nomount” the database to be created. Start the spooler to create a
log file for the db creation and execute the db creation script. Check
the spooled log file for errors.

Assuming that there aren’t any errors in the spooled log file
and the database is successfully created, a data dictionary must be
created. Catalog.sql and catproc.sql must be run using the Oracle
supplied catcon.pl program. The data dictionary and all the PL/
SQL packages, etc. need to be built in the root database as well as
the seed database. If catcon.pl isn’t used an “ORA-00942: table or
view does not exist” error will result when creating pluggable
databases. The seed database is read only and catcon.pl is needed
to build the data dictionary in the seed. Assuming that you wish
the “catcon.pl” script to write log files to the “/home/oracle/ora-
data/CDB_Seed/” directory, the commands to build the data dictionary with catcon.pl are
as follows:

Listing 3 - Using catcon.pl to run catalog.sql, catproc.sql21

$ export PERL5LIB=$ORACLE_HOME/rdbms/
21  /admin:$PERL5LIB;

$ perl $ORACLE_HOME/rdbms/admin/catcon.pl
22  -n 1 -l
23  /home/oracle -b catalog $ORACLE_HOME/rdbms/admin/
catproc.sql;

Creating Pluggable Databases

Once a container database has been created it is possible to
create pluggable databases within it. Remember that a container
database is a meta-database combined with a read only seed data-
base. The root database (CDB$Root) should not contain any data
other than meta data in the data dictionary. A pluggable database
can be created from the read-only seed database (CDB$Seed) that
was created when the CDB was created, or from an existing PDB,
or by creating a PDB from a non-container database, or from a
database created in another container database and subsequently
unplugged from it.

All of the databases within a container database share boot-
strap files (control files, init.ora, pfile.ora), the redo logs, the
undo tablespace and optionally may share a temporary tablespace
with the container database. Each pluggable database within the
CDB does have its own System and SysAux tablespace. All of the
databases share the processes used to run the databases. In other
words, there is one set of smon, pmon, dbwr etc. processes that are
common to the root, seed and all the pluggable databases. This
does restrict the “alter system” commands that can be performed by
a local SYSDBA in a pluggable database. The initialization param-
eters that may be changed are listed in the v$system_parameter
view where the “is_pdb_modifiable” column is set to “TRUE”. Other
system commands that may affect individual pluggable databases
are listed in the DBA Administrators Guide and include operations
such as flushing the shared pool, the buffer cache, etc23.

All pluggable databases created are cloned from other plugg-
able databases. Since they share the container database’s initial-
ization file another init.ora is not required.

The PDB being cloned must be in read only mode. If the PDB
is open in read write mode,24 close it first (alter pluggable database
<pdb> close); then open it in read only mode (alter pluggable data-

---

20 “SORACLE_HOME, and SORACLE_SID must be set. SORACLE_HOME/bin
must be in your $PATH and SORACLE_HOME/lib in your $LD_LIBRARY_PATH.

21 These are long command lines. I did not put in line continuation char-
acter is this listing. The “$” is a Linux prompt and the line continues until the next
prompt.

22 “perl catcon.pl” will list the options or see Chapter 40 in the Oracle
Database Administrators Guide

23 The commands include flush shared pool, flush
buffer cache, enable/disable restricted
session, set use stored outlines, suspend,
resume, checkpoint, check datafiles, regis-
ter, kill session and disconnect session.

24 “Select name, open_mode from v$pdb$” shows the status of the PDB.
Creating a PDB from the Seed (PDB$Seed)

In order to create a pluggable database from the seed database you must log into the root database as SYSDBA. The files associated with the seed database (PDB$seed) are copied to another location and then associated with the container database using the create pluggable database statement.

You must specify a local administrator for the PDB in the create pluggable database statement. The statement creates the administrator as a local user in the newly created PDB and grants the PDB_DBA role locally to the administrator.

Two options are available to grant privileges to the PDB_DBA user. You may grant privileges or roles after PDB creation or you may include the roles you want to grant to the local PDB_DB in the SQL statement used to create the PDB. If you wanted to grant this PDB_DBA user some set of privileges through roles during PDB creation, the create statement should look like the following:

```sql
Create pluggable database <pdb>
admin user <admin_username> identified by <password>
roles = (<role_1>, <role_2>, <role_n>);
```

This statement will create data files for the PDB as long as either Oracle Managed Files is set or the PDB_FILE_NAME_CONVERT initialization parameter has been set. You may also use the FILE_NAME_CONVERT clause in the create pluggable database statement if you have not set PDB_FILE_NAME_CONVERT or are not using Oracle Managed Files. The final way to specify locations for data files is by using the path prefix clause in the SQL create statement to specify the absolute paths of files associated with the PDB. If Oracle Managed Files is set the path prefix clause will be ignored.

You may also specify a default permanent tablespace in the create PDB statement. It’s semantics are exactly the same as they are in the create database statement including the storage parameters. Any tablespace created with the create pluggable database statement may also have storage limits specified for it in the create statement.

After the PDB is created, it is in mounted mode. It must be opened in read-write mode to access it (alter pluggable database <pdb> open read write). You must also configure Oracle Net so that local PDB users may access the database.

```sql
pdb open read write;)
```

You must also configure Oracle Net so that local PDB users may access the database.

Creating a New PDB by Cloning an Existing Pluggable Database

Creating a new pluggable database from an existing pluggable database is much the same as creating a PDB from the seed database. Connect to the root container as SYSDBA and make sure the PDB that is to be cloned is in read only mode.

The major difference between cloning the seed and cloning an existing PDB is adding a “from clause” to the create pluggable database statement. When the from clause is missing the new PDB is created from the seed database.

A new PDB may be created from a PDB in the same container, from a PDB on the same host but in a different container or from a PDB in a container database on a remote machine. In the case of a PDB outside the same container, if one doesn’t already exist, a database link to the other pluggable database must be created.

All of the same options exist as they do with creating a PDB from a seed with the addition of the “snapshot copy” option. The basic statement looks like the following:

```sql
Create pluggable database <pdb>
from <existing_pdb>@[database_link]
admin user <admin_username> identified by <password>
roles = (<role_1>, <role_2>, <role_n>);
```

Creating a New PDB from a Non-Container Database

It is only possible to create a pluggable database from a Non-Container database via SQL*Plus if the non-container database to be converted is already version 12c. The first step is to set the environment to access the non-CDB database. Shutdown the source DB if it is running in read write mode and open the non-CDB in read only mode (startup open read only).

Once the non-CDB is running in read only mode an XML file can be generated that contains the non-CDB metadata describing the structure of the non-CDB database using the DBMS_PDB DESCRIBE procedure. The XML file that is generated is the same as the file generated when unplugging a PDB from a container DB and allows the non-CDB to be plugged into the destination container DB. Once the non-CDB has been described it can be shut down. See Listing 4 for a sample script to generate the XML description file.

Listing 4 – Sample script to create XML metadata to plug a non-CDB into a CDB

```sql
begin
    DBMS_PDB.DESCRIBE(
        pdb_descr_file => '/u01/workdir/non-cdb12c.xml');
end;
```

Before you plug the non-CDB into a container database, it

---

25 It’s as easy as adding an entry to the tnsnames.ora for each new PDB.
26 Pardon my flair for the obvious, the seed database is also a pluggable database in case you wondered.
27 There aren’t any grants associated with this role.
28 It might be the policy at your workplace that only common dba users should be able to administer pluggable databases so you might leave the local pdb_dba user without privileges and use common dba users to administer the new PDB.
29 It’s as easy as adding an entry to the tnsnames.ora.
30 Creating a PDB in the source CDB, unplugging the new PDB and plugging it into the destination container is a much better practice if the source and destination containers are on different physical hosts.
31 If the files are stored using Oracle ACFS or Direct NFS Client storage you may specify the SNAPSHOT_COPY clause to clone a PDB using storage snapshots.
32 If the source DB is not version 12c or otherwise incompatible your choices are: upgrade the non-CDB database to Oracle version 12c, create a new PDB from the destination PDB$Seed DB and use Oracle Data Pump to export and re-import the data from the source into the newly created PDB or use Golden Gate Replication to move the data from the source database into the DB created from the seed.
must be checked to see if it has the same “endianness”\(^{33}\), the same set of options installed, and have a compatible character set with the CDB it will be plugged into. Compatibility is checked by logging into the root database of the destination container and using the DBMS_PDB.CHECK_PLUG_COMPATIBILITY function on the XML description file to check. Listing 4 shows an example script to check PDB compatibility.

Listing 4 Sample script allowing a destination container DB to check a source DB's compatibility with the destination CDB.

```
SET SERVEROUTPUT ON
BEGIN -- returns true or false
DBMS_OUTPUT.PUT_LINE('The Non-CDB is compatible: ' ||
    DBMS_PDB.CHECK_PLUG_COMPATIBILITY(pdb_descr_file =>
    '/u01/workdir/non-cdb.xml');
END;
```

Once the XML metadata file is generated and compatibility has been checked the PDB can be created. If the source database files can be left at their original location then the NOCOPY option may be used with the create plugable database statement. If you wish to move the source DB files to a new location, the MOVE option can be used along with the file_name_convert clause listing the original files and their new locations. If the COPY option is used, the source db files are copied to a new location. COPY is a handy option since it allows making multiple copies of the source db at any time by using the CLONE option along with create pluggable database to make multiple copies of the same database as if it was a seed\(^ {34} \). See Listing 5 for a sample script to create PDBs while moving or copying a non-CDB files to a new location. See Listing 6 to NOCOPY the source files and see Listing 7 to create a clone from the source DB files.

Listing 5 – Sample script to plug a Non-CDB into a container database while moving or copying files.

```
CREATE PLUGGABLE DATABASE <new_pdb_name> USING '<non_cdb.xml>'
  [MOVE|COPY] FILE_NAME_CONVERT = ('/u01/NEW_PDB/');
```

Listing 6 – Sample script to plug a Non-CDB into a container database keeping the source database files in the original file location.

```
CREATE PLUGGABLE DATABASE <new_pdb_name> USING '<non_cdb.xml>'
  NOCOPY;
```

Listing 7 – Making a copy of the source DB files to CLONE the source database.

```
CREATE PLUGGABLE DATABASE <new_pdb_name> AS CLONE
  USING '<non_cdb.xml>'
  [MOVE|COPY] FILE_NAME_CONVERT = ('/u01/MYSOURCEDB/','/u01/NEW_PDB/');
```

33 The way that the bytes of a data word are ordered in memory.

34 Making multiple copies of a PDB after you import is a technically easier option although if you are importing PDBs into production, cloning from an XML description and the DB files does have its advantages.

The last option to be aware of is the source_file_name_convert clause. If the source database files have been moved from their original location they can still be plugged into the destination container. For example, if the source database resides on another machine and the files were copied to the destination machine, use the source file name convert clause to plug in the non-CDB. See Listing 8 for an example.

Listing 8 – Using the source_file_name_convert clause to plug the non-CDB into the destination container.

```
CREATE PLUGGABLE DATABASE <new_pdb_name>
  USING '<non_cdb.xml>'
  SOURCE_FILE_NAME_CONVERT = ('<current_location>', '<old_location>')</n  NOCOPY; -- db files to remain at <current_location>
```

The final conversion step is to connect to the new PDB and run the noncdb_to_pdb.sql script in the $ORACLE_HOME/rdbms/admin directory to complete the conversion. Since you are running this script in only one PDB there is no reason to use catcon.pl. Once the noncdb_to_pdb.sql script finishes, “alter pluggable database open read write” will complete plugging the source database into the destination container and open it for use.

Creating a New PDB from an Unplugged PDB

The last type of PDB creation I’ll illustrate is unplugging a PDB in one container and then creating a PDB by plugging it into another CDB. It can be argued that this is moving a database rather than creating one unless the clone option is being used.

The first step is to log into the root database of the source container as SYSDBA and unplug the database to be moved from the source container. The PDB must be closed (alter pluggable database <pdb> close) and then the unplug clause used to unplug the PDB into an XML description file. “Alter pluggable database <pdb> unplug into '/$myWorkDir/<pdb>.xml'” unplugs the PDB. In reality most of what this does is create an XML description file which along with the other database files will allow it to be plugged into another container DB with the create pluggable database statement. It also changes the status of the PDB in the source CDB to “unplugged”.

At this point the PDB is unplugged from the current container but it is still associated with the source container and the CDB control files still contain references to the unplugged PDB. The PDB being moved can still be backed up with RMAN in the source container.

The next step is to drop the PDB from the source CDB while keeping the datafiles with “drop pluggable database <pdb> keep datafiles”. The PDB must be dropped from the source CDB to plug it into a destination container even if the destination is the same container it was unplugged from.

At this point all that you need to do is check it’s compatibility with the DBMS_PDB.CHECK_PLUG_COMPATIBILITY procedure and if it is compatible then use the create pluggable database statement to plug the PDB into the destination container. The same options apply when using create pluggable database with a PDB that has been unplugged as they do when creating a PDB with a non-container database. The last step is to open the new pluggable database in read write mode.
Summary

This article shows how to create container, pluggable and non-container databases with SQL*Plus. Container databases are meant to serve as containers for pluggable databases. Pluggable databases may only exist when associated with a container database. PDBs share many physical structures as well as the processes that run them with the container database. Non-container databases are free standing databases similar to the type of databases previously associated with Oracle databases.

Container and Non-Container databases are created with the "create database statement. The creation process for container databases adds the "enable_pluggable_database" clause to the init.ora and the create database statement.

Pluggable databases may be created by cloning another pluggable database or a Non-Container database using the "create pluggable database statement". The database cloned may be the seed database created when a container database is created, another pluggable database either in the same or a remote container, a PDB unplugged from another container or by moving a non-container database into a container database.

Note that after you create any new pluggable database attention should be paid to the effect on the CDB it is associated with so that its creation does not adversely affect the performance of other databases in the container. It is good practice after adding a pluggable database to consider tuning the container database to re-optimize its performance.
Scholarship Recipient

Congratulations to John Crews, recent recipient of the RMOUG Stan Yellot Scholarship Award. The Scholarship Fund started in 2001 to encourage future IT professional in their efforts to broaden their knowledge. It was renamed in 2007 to honor Stan Yellot. RMOUG is grateful to assist students in their education.

John Crews is thrilled and humbled to receive the Stan Yellott Scholarship. John comes from a family with a strong legacy of military service and engineering endeavors. He is named after his grandfather, John R. Crews, who was a Medal of Honor recipient in World War II. John’s brother, Joseph, is a Systems Engineer in the US Navy and his father, Dr. Mark Crews, is a retired US Air Force Colonel, Electrical Engineer, and continues to support our nation’s GPS system through civilian employment. John began software coding in 2011 at Pine Creek High School, under the capable and encouraging teaching of Ms. Denise Gardiner. In 2013, he attended RMOUG’s Training Days conference as part of a Pine Creek High School class trip sponsored by Stan Yellott. The RMOUG conference both inspired and reinforced John’s drive to pursue software coding as a career path. In school, John excelled in C++, SQL, and Java courses. He has been accepted into the College of Engineering at the University of Colorado, Colorado Springs (UCCS), and is pursuing a degree in Computer Engineering. He views computers and machines not just as means to a living, but as a real way to make the world better through connectivity and learning. John is extremely honored to be a recipient of the Stan Yellott Scholarship, in addition to carrying forward the technology legacy of Jeffrey Meacham (2014 Jeffrey A. Meacham Technology Scholarship recipient).

From The Editor

Transition is a natural part of our lives. Yet there are times when transition is a parallel event with other transitions. How we deal with transition expresses itself as stress and emotions. I’ve heard of stress defined as your body’s response to anything that disrupts your normal life and routines. By this definition, transition and stress go together.

I hope to pass along some tips on transition to help others who may be going through some transition of their own. As the new RMOUG newsletter director, it is my responsibility to find anyone interested in writing an article for this prestigious magazine. That task has proven more challenging than I expected. I did not want to send out a message to every member, yet that seems to be the way this works. Members will be hearing from me on a more frequent basis. Find something you understand and share it with our group. I truly believe we all have something to share. The articles don’t need to be technical, although that is the main focus of SQL>UPDATE. Transition #1.

Last year, I finally became an Oracle Certified Professional. Studying for this certification was interesting and I found the most relevant questions – the ones that were literally exact wording from the test, were the ones I discovered the day before the test. Having completed all my study material, I searched one last time for any questions I could find. Now I am looking at upgrading my certification to 12c. Transition #2

On December 17, 2013, the company I work for was bought, eliminating my specific job. They were generous in regard to this transition, asking me to stay through the completion of the merger process. I began my new job search immediately with the stipulation that I could not begin work until June 1, 2014. In the current technical world, companies are looking for someone to step in and solve problems immediately, so letting them know you aren’t available for months simply does not work. Then I had my work agreement extended (voluntarily) until August 1. So I have been practicing my interviewing skills and learning that there is no standard interview process. Some places are very technical; some are more about how your personality fits the team. I was told at one interview that some of my stories of past experiences were simply too fantastic and I must have been simply saying what they wanted to hear. Transition #3.

This experience and the idea of transition has me thinking about my career path and where I am heading. I tend to be an active administrator of data, walking into the offices of the people I support to get a better understanding of their issue or concern than I could sitting at my desk. I have always wanted to be part of a larger solution, finding the root causes of company misdirection and not just from a technical perspective. I’ve managed teams of administrators and developers before and I find I am leaning more that direction. So how exactly does a database administrator transition into a project manager? There’s that word again. Transition.

So, help me in my transition and write an article for our magazine. Technical, functional, or otherwise, if you are imparting knowledge to our band of Oracle warriors, we would love to hear from you.

Vincent Giasolli
Real-Life Recovery: Perspective, Preparation and Performance

by Daniel W. Fink - 7S Consulting, Inc.

In an anonymous office building downtown, an Oracle database administrator checks the log files to verify that the backup processes ran properly last night. Then he grabs a 2nd cup of coffee and moves on to the day’s tasks, emergencies and assorted events. Users, managers and developers are concerned with data accuracy, proper security and blinding performance, not in the mundane task of testing recovery. Little does he or the company know that the past month’s backups are useless. Since the recent operating system update, the tape drive, the only tape drive on site, writes without error, but cannot read a single bit that is on the tape.

A few miles away, 2 DBAs and a System Administrator are completing a 12-hour system and database recovery. No one has slept in over 36 hours. Mistakes were made, but the errors were caught well before the right pinkie was poised over the <ENTER> key. When the final command was issued, the three felt suddenly refreshed…success is a great stimulant. For the 3rd time in as many months, they have successfully performed a recovery with no perceivable loss of data. The Operations Manager walks in with a pot of coffee and a piece of paper. “Pretty good, no data loss,” she says, “but you could have been done 2 hours ago. I’ll grade it a B+. Let’s go for an A next time! Go home, get some rest and we’ll see you tomorrow.”

These two tales illustrate several important points regarding Oracle Backup and Recovery. Most sites focus on backups and other issues before addressing recovery. Single points of failure are shortcuts to disaster, both business and career. The only measure of a successful backup plan is how completely recovery can be performed. The time to practice a recovery is only measure of a successful backup plan is how completely recovery can be performed. That is also the worst possible time to find out that the backup process is flawed, the SYSTEM tablespace file cannot be read from the tape.

Perspective

Most Oracle documentation gives equal billing to backup and recovery. The routine task of setting up, monitoring and maintaining backup processes consumes valuable resource time. Once the process has been established, other issues, such as security, tuning and troubleshooting move to the top of the DBA task list. Usually, a crisis is the time that the recovery process is performed. That is also the worst possible time to find out that the backup process is flawed, the SYSTEM tablespace file cannot be read from the tape.

This situation is a result of the attitude that the DBA is responsible for performing database backup. The actual respons-

sibility is to restore or recover the database to the point in time and within the downtime window determined by the business needs.

Rather than the backup process dictating the recovery options, the recovery requirements should drive the backup process. A backup strategy is determined by the answer to a single question, “What is the best method of backup and archiving that meets all of the business needs of data loss, downtime and cost?” As with many decisions, the question appears to be simple, but is complex when defining ‘best method’, ‘business needs’, ‘data loss’, ‘downtime’ and ‘cost’. In an attempt to define these variables, the Cost v. Lost Revenue Model is the best method to assist in answering this question.

Restore, Recover, Rebuild

To restore is to return to a former state. In database terms, this means to return the database file(s) to a previous point in time determined by copying the file from backup to online media. The restored database or particular file may be missing transactions. Restoration is the first step and is required before database recovery can be performed after a media or other similar failure. If the database is not running in archivelog mode, the restore process is probably the opened state of the database. Without archive logs, recovery can only be accomplished to the date of the last backup.

To recover is to return to a normal state. In database terms, this means to return the database to a point in time determined by the situation. It may be a complete or incomplete recovery, depending upon the circumstances. Files are restored, and then archived redo logs are applied to bring the database to the desired state. Some transactions may be lost, but they are usually few.

To rebuild is to recreate a specific state. In database terms, this usually involves reusing the data and process that populated the database in the most recent iteration. In some environments, particularly data warehouse or nonproduction, the raw data exists outside of the database. There may be no actual transactions since the last load therefore nothing is lost.

Base Knowledge and Training

Transaction Architecture. The reason for a database is to support transactions, whether they are entering a $9.99 cable order or running a 3-day Regional Sales data warehouse query. The understanding of how a transactions works, and what happens when it does not work, are the cornerstone to database knowledge.

Recovery Structures and Processes. A DBA must be able to answer the question “What is the difference between a rollback segment and redo log?” in 25 words or less. Know the recovery structures, what they do and how to recover from the loss of each one.
Points of Failure

“Never have a single point of failure’ is the Oracle Backup & Recovery Maxim. The basic foundation is that a single failure should never prevent a backup or recovery process from succeeding. If day-old archive logs are deleted after the cold backup, the backup tape is a single point of failure. If the logfiles are unable to be restored, the previous backup cannot be used to recover the database because the archived logs are unavailable.

Identify the ‘show stoppers’ and have at least 2 options for bypassing. A single bad backup or command should never be the cause of a failed recovery. Taking an extra 2 hours to recover a lost datafile from a 2-day-old backup is much easier than explaining why yesterday’s sales were lost because the archived redo log was lost.

Prevention of failures is a proactive approach to recovery. Recovery from a dropped table is possible, but implementing and enforcing database security, access and maintenance procedures can prevent the situation. Using an appropriate RAID configuration can minimize the impact of disk loss. There are expenses associated with prevention, but they may be minimal compared with actual recovery costs. Even if proactive solutions are implemented, there will still be areas of weakness. While certain RAID configurations will protect against the loss of a single disk, they cannot protect against the delete command.

Failure Prone Components

Personnel The most failure prone component is the person issuing the commands. Document the steps required for each recovery type. When documenting, assume little, if any practice. The recovery manual should address the areas that pose the greatest chance of total loss of service. Use signposts like ‘Proceed with Caution’, “Stop & Review”, etc. and decision trees. Update the manual periodically, at least at every major change. If you are using Oracle Support assistance, make certain that you are talking to someone who has performed recoveries. Don’t be afraid to ask!

Redo Logs (online, offline and archived) The loss of a single redo log can result in total recovery failure. This is one reason that Oracle recommends using mirrored redo log groups. Using hardware level mirroring is not sufficient, you are protected from media failure, i.e. disk loss, but you are not protected from the delete command. Archived logs should exist in their natural state on at least one backup tape. Once there, they may be compressed and kept online for several days, and placed on several other backup tapes. This strategy offers good protection from loss and assists in recovery speed by having the logs online. Although rare, the compression/uncompression cycle may result in corrupted files, so use with caution. If disk space allows, have at least two uncompressed archive log copies on different tapes.

Control Files If all control files are lost, Oracle offers several methods for performing recovery. However, these tasks are risky and easily avoidable. Another Oracle maxim is to have 3 copies of your controlfile, because a corrupt controlfile will always (well, almost always) be copied over a good controlfile. If you have a 3rd controlfile, the erroneous copy will still happen, but only you will know...the 3rd copy is used to save the day and your job.

Initialization Parameter Files These files are often overlooked. As with controlfiles, their loss can be overcome, but at a substantial effort. Data is not lost, but downtime is increased. A copy of all parameter files and a dump of v$parameter add several layers of backup.

System Tablespace datafile(s) These datafiles may be the most important in the database for it is the blueprint. Without the data dictionary, Oracle is blind to all of the data that is residing in the datafiles.

Data Tablespace datafile(s) Under the OFA standard, data and indexes are separated into distinct tablespaces. This allows backup and recovery plans to be flexible. If the system and data tablespace datafiles are recoverable, all other tablespaces can be built from scratch, if sufficient documentation exists. For example, indexes can be recreated, if the most current DDL scripts are available.

Rollback Segment Tablespace datafile(s) Although these tablespaces are used by transactions to provide transaction recovery and read consistency, the loss of this tablespace is not necessarily a ‘death sentence’. The key factor is the state of the database at the point of the backup. If it is in consistent mode, (i.e. no uncommitted transactions) there should be no rollback entries. As such, the segments can be dropped and rebuilt without data loss.

Index Tablespace datafile(s) As stated above, if the most current DDL scripts exist, indexes can be recreated from existing data. There is a substantial recovery performance penalty incurred as the indexes are being rebuilt, but no data loss occurs.

Temporary Tablespace datafile(s) Once the database is shutdown, the temporary segments are ‘clean’. If the temporary tablespace(s) are lost, recreation is a fairly simple process. As with indexes, there is a recovery performance penalty, but no data loss occurs.

Misc Files File maps, object recreation scripts, backup scripts and other assorted files should be part of the backup process. Performing a media recovery is greatly simplified if an up-to-date tablespace-file-device map is available. While these files are not required for recovery, they can shorten and simplify the recovery process.
Preparation – the Cost v. Lost Revenue (CLR) Model

In a perfect world, the CIO would hand the DBA and SA a blank check for backup hardware/software and provide an unlimited operations budget. In the real world, there is a balance between the costs associated with backup and the benefits of recovery. Although the business needs drive the Cost-Benefit Analysis, it is the responsibility of the technical staff to educate the users and support the decision-making process. It is not the responsibility of the technical staff to make the decision.

The basic concept is to compare the cost of backup and recovery with lost revenue. This is balanced so that the frequency of backup v. the frequency of recovery is appropriately weighted. In a production environment, a downed Order Entry system can be quantified. In a development environment, the quantification is related more to lost Developer/User time.

The model should be generated for 3 scenarios – worst, best, and anticipated. The anticipated case should be somewhere between worst and best. Best case should not be a fantasy, assuming some level of recovery requirements.

There are five basic steps to the CLR Model process:

1) Educate the decision makers

Discuss issues in clear, nontechnical terms that are clear to everyone. The first step is to lay the foundation by educating the decision-makers in basic knowledge of backup and recovery. Although an Accounting Manager does not need to understand the intimate details of archive logging, they do need to understand that transaction recovery is not possible without it...and what the business implications are of adopting this strategy.

2) Assign costs to each resource.

This phase is primarily devoted to assigning actual or opportunity costs to each resource. The resources are hardware, software, and people. The backup process will consume disk space and/or tapes, CPU cycles and memory on the host platform. Backup software must be purchased or written. Administrators of various experience levels must monitor backups, perform tests, document and perform the occasional recovery.

For hardware and software, acquisition and operation costs can be quantified or estimated. The operation costs are defined within a specific scope of time. The actual time is not important, but it must be consistent. Hardware operation costs for a year are not comparable to backup operation costs for 6 months.

For personnel, the actual or opportunity cost is determined. For most production environments, personnel cost is actual, i.e. the sum of salary and benefits or hourly charges. For development environments, the cost may be classified as opportunity, i.e. the hourly rate that the client is being charged.

Although they are not assigned at this point, the intangible costs should be discussed. These costs include impact on project timelines, goodwill with users, management and technical personnel. If a project is approaching a critical junction, the cost of downtime may escalate dramatically. Users may tire of experiencing excessive downtime; Managers may tire of user complaints; Developers may tire of reloading the past month’s data; Administrators may tire of 36-hour days. The short-term impact may be less money spent on the systems, but the long-term impact may be frustration, high turnover and added costs.

3) Assign costs to each backup strategy

There are three types of backup strategy: hot, cold with archiving and cold without archiving. A cost for each scenario is calculated in two areas: fixed and variable.

Fixed costs are the one-time costs for each step during the operational period. Software needs to be purchased or written once. Disk space required for Archived Logs or backup datafiles is another one-time investment. These costs are incurred regardless of the frequency of backups, assuming the frequency is greater than 0.

Variable costs are incurred each time the backup process occurs. Logfiles must be monitored, tapes must be written and stored and scripts must be maintained.

At this point, the first business decision is made. The anticipated frequency of the database backups is determined. This decision is very preliminary, it is not set in stone. The less frequent the backup, the more costly the recovery, but a final decision is premature. It is important to use the preliminary decision as a baseline. Once the recovery costs are determined, the type of backup can be revisited.

Another alternative is to complete this phase of the model for each type and frequency of backup. While time consuming, it may be simpler and quicker when busy schedules and diverse audiences are involved.

4) Assign costs to each recovery scenario

The most common types of recoveries are object, file and disk, each with a distinct strategy. Cost determination is common among all of the strategies, and is slightly different from determining backup cost.

The first area of recovery cost is practice cost. The major fixed expense is for a practice platform. An ideal situation is to have a ‘sandbox’ for System and Database administrators to test upgrades and recoveries. In many situations, an older development platform may be adequate. Added to the cost is the variable personnel cost for each practice.

The second area of cost is restoration. This includes the time to detect and correct/bypass the cause of failure. Then the backup media is retrieved and objects or files are restored. If archiving is being used, only the affected object or file needs restoration, which can significantly reduce the restoration time.

A final, but optional, area is the cost of recovery. If archiving is being used, Oracle recovery processes can be used. This is usually the quickest method of recovery. Other methods are transaction reentry and data reload, which are both time and personnel resource intensive. For a small decision support system, it may be more efficient to recreate the database from tested scripts and reload the previous period’s data.

The best method for estimating recovery time is to actually practice each step. An alternative method is to use estimations for each step. Regardless, a qualified guess is better than nothing, but remember that these decisions are among the most important that may be made in your career.

Support the business user(s) in making an educated decision. Once all of the costs are determined, the process of finding an acceptable balance is begun. Unless the company can afford hot-standby, fault-tolerant, fail-over systems, compromises must be made. Simply put, it is time to make an educated, documented decision that is supported by the business. The decisions that are made during this step may result in lost data, missed sales, overtime and tough explanations to management. If it is determined that the decision requires additional resource, such as disk drives or a tape library, they can be purchased before they are needed. Recovery planning is part of the system design phase.
Although it may not be easy explaining to management to invest $100,000 in a system that is rarely used, it is better than explaining why $1,000,000 in sales were lost yesterday because the Order Entry system is not archiving transactions.

**Performance**

Oracle database recovery is one area where the line between success and failure is clear. If the database is recovered according to the defined business requirements, the recovery was successful, if not, the recovery was a failure. Trying to successfully recover is not acceptable. In the words of Yoda, “Do or Do Not. There is no Try.”

The only valid method for testing a backup process is to perform a test recovery. This test should be repeated on a regularly scheduled basis and after any major change, such as a new tape device or system upgrade. The time to find out that the tape drive is not functioning properly or that a new database file is not being backed up is during a practice run, not a live recovery situation. Practice runs will also expose gaps in procedures, documentation, training and, most importantly, confidence.

**Practice, Practice, Practice**

Practice recoveries are the only method for gaining confidence in the process and exposing weaknesses. Being able to quote chapter and verse from recovery theory is not a substitute for drawing upon actual experience, especially at 4:00 in the morning after a 20-hour day. At that time, you may be one keystroke away from success or failure. It is also the time when your mental faculties are at their most vulnerable.

Mistakes during a practice recovery cause no harm. In fact, they are a tremendous learning experience. If you do not have the time and training to perform a recovery 100% right the first time, there may be no next time...at least not at the current company.

Practice will expose the ‘bridge burning’ steps. These points are critical decisions where returning to a previous step is difficult or impossible, unless certain precautions are taken. Copying a backup datafile in place of an existing datafile is such a step. These steps must be documented and well understood. It may be appropriate to backup a ‘bad’ datafile prior to restoration. This allows for a retreat should it be needed. Treat these situations as ‘show stoppers’.

Consider tools used for the recovery process, such as Recovery Manager or other 3rd party products. If the business uses a tool, it needs to be part of the practice. You must also plan for the possibility that the tool is unavailable, always practice manual forms of recovery. Tool availability is not a substitute for basic foundational knowledge.

Practicing recovery is not a minor expense, but it pales in comparison to losing a substantial amount of data for a production environment. It requires professional dedication to the business. As recoveries are practiced, knowledge and skill are developed and confidence is increased. In turn, this may assist in redeploying the backup and recovery strategy.

**Executing the Plan**

1. **Stop panicking.** It is imperative to get into the proper frame of mind. Taking an extra 15 minutes to calmly discuss the possible alternatives may make the difference in a failed recovery attempt and an additional 15 hours of downtime.
2. **Determine scope and cause of failure.** The type of failure will have a serious impact on the types of recovery to be considered. If the failure will be repeated shortly after the recovery is performed, the recovery is wasted. If a table has been dropped, performing a complete recovery will return the database to the point AFTER the table was dropped.
3. **Correct/Bypass failure.** When a disk fails, the datafiles should be restored to another disk or the disk should be replaced. If the failure is not corrected or bypassed, a recovery may be wasted.
4. **Identify Plan of Attack.** Once the failure is understood and corrected, the recovery options are defined and a plan is determined. If a datafile for an index-only tablespace has been damaged, it may be more efficient to drop the tablespace and rebuild the affected indexes. Depending upon the plan, the next step may not be required.
5. **Restore affected data.** After failure correction, the process of restoration can begin. A tape or other backup media is retrieved and the affected data and log files are restored to the system. If redo log files are not being archived, the complete database must be restored.
6. **Perform recovery.** The actual recovery strategy that was determined in the plan is performed. In the above example, this may involve dropping the tablespace, recreating it and then rebuilding the indexes and recreating constraints. Depending upon the backup plan, there may be substantial resource requirements, especially if the data must be reloaded or transactions reentered. If time allows, a full cold backup should be performed after recovery but before the database is opened for general use.
7. **Postmortem.** Debrief, document and determine improvements. Determine if the failure situation could be prevented. Losing data and incurring downtime due to a preventable failure is rarely acceptable in a business critical system, unless the business requirements have accounted for and accepted this possibility.

**Conclusion**

Among the myriad of DBA tasks and responsibilities, the most important to the business is the ability to properly recover from a failure. There are many factors that determine the type and scope of recovery, primarily the balance between what the business can afford to spend and what it can afford to lose. These decisions are not the responsibility of the DBA or SA; rather, they are to be made by key members of the user community. The technical staff functions as a support organization by educating the users and assisting in the decision-making process using a Cost v. Lost Revenue Model.

Once the business has determined the backup and recovery needs, the technical staff becomes responsible for insuring that these operations are properly executed. As a technical administrator, you are the greatest strength and greatest weakness for the recovery process. Training and practice are the paths to success. The recovery process is too critical to depend on less than 100% effort and ability.

Daniel Fink is an Oracle DBA with experience dating back to Oracle 7.0 Parallel Server on OpenVMS, the same year he attended his first RMOUG Training Days at the US West Training facility in Lakewood. His recent experience includes Exadata, RAC, ASM and other assorted buzzword technologies. When not writing short biographies for articles, he can be found somewhere in the mountains.
With the popularity of social media, more focus has been placed on not just company brands, but personal brands. A personal brand is your online identity on the internet and to create a powerful personal brand requires thought, persistence and a desire to learn about what it requires to be successful with the world of social media and the internet.

Most will tell you that it simply relies on you being out there, yet knowledge and tools can assist in making it easier. You should have a clear idea of the image you wish to present to the public, know the value you bring to the community that you are involved in and how you wish to grow as part of it. As my personal brand, DBAkEvlar, (@DBAkEvlar, http://about.me/DBAkEvlar) is based off my technical identity, it provides a synchronized set of profiles and clear representation of who I am in the technical community no matter what platform you view.

**Personal Brand Categories**

To build your own personal brand, you should first consider what is unique about you and what you are passionate about. These two traits can then be used to build an identity that will classify who your followers will be by interest and similarities.

After building an initial profile, you have to think about what type of involvement in social media you will have. There are four common categories of social media users:

- **Networkers:** Professional contacts are everything
- **Socializers:** Interaction and communication, often social in nature, less professional.
- **Lurkers:** Monitor social media, but rarely post anything.
- **Broadcasters:** Send out information, interact less vs. the amount of data broadcasted.

Outside of social media, you then have the following:

- **Authors:** Write articles and/or books.
- **Contributors:** Contribute only to sites owned by other groups/individuals.
- **Content Providers:** Simply push out content from other providers, nothing original.
- **Bloggers:** Post on their own site, original content.

My DBAkEvlar brand is a combination of a number of these, as I'm heavily involved on the internet compared to most users. For social media, I'm considered a networker and broadcaster. Outside of social media, I fall into all four categories, as I produce my own blog and magazine, write for other publications, push content and contribute to other's sites as well.

Some of these categories, like the social media ones, are part of online scoring, most well-known on Klout (www.klout.com). The scoring is a bit skewed, so I've never put too much thought into klout scores, but I do like the site for the other features it provides as a social media tool, (which we’ll discuss a little later on...)

Making a decision concerning your social media interaction will define the value of your personal brand. Someone who chooses to simply “lurk” and then invests their own content in blogging only will have less of a reach than someone who interacts with other social media users who will then want to follow their blogs. A personal brand’s reach will also be increased if articles are contributed to another site. Someone appreciating the article’s content would very likely follow the author’s blog and social media content.

**Social Media Platforms**

The most popular platforms for social media are shown below, along with links to each site.

- Twitter, http://twitter.com
- Facebook, http://facebook.com
- Google+, https://plus.google.com
- Instagram, http://instagram.com
- Pinterest, https://www.pinterest.com
- Vine, https://vine.co
- LinkedIn, https://www.linkedin.com

These are each used in different ways and to reach different audiences. We’ll start with the last one first, as it is unique in that it is exclusively professional in its audience.

**Professional Audiences**

LinkedIn

LinkedIn is for companies, professional groups, organizations, (both for and not for profit) and professionals to connect, network and provide content. LinkedIn, (http://linkedin.com) is an essential networking site.

Building a professional CV, (curriculum vitae, i.e. resume) is important, using a clear and professional headshot for your profile. All connections in your network should be as involved in your profession or assist your profession as much as possible. Keeping your profile up to date at all times is in your best interest for two reasons:
Anyone looking at your profile will have up to date information on you.

LinkedIn sites that are updated infrequently usually indicate people looking for a new job. By keeping your LinkedIn site up to date, you avoid the recruiters thinking you are looking for new work.

I would also recommend joining groups that interest you and if you are offered to post articles, only write ones that are professional in nature. Nothing personal should be included on LinkedIn.

**Personal and Professional Audiences**

All other social media platforms are used for both professional and personal, but they are used to reach different audiences and have different advantages and limits.

**Visual**

Instagram and Pinterest are both used to upload pictures with limited captions. Vine is used mostly for gifs and short video. As I am a techie, I use Instagram and Pinterest very rarely and only for photos at conferences or impersonal objects. If you are in marketing, art or any graphic design, I could see how any of these would be very valuable social media platforms for your profession.

Twitter, Google+ and Facebook can be for professional and personal. I choose to keep my Facebook for personal and Twitter and Google+ for professional. This helps keep a definitive line for family and friends. I also use Google+ along with YouTube to push videos of my user group involvement and to advertise for conferences. I push content to Google+ as I do Twitter and Facebook, but there is a limitation with Google+. One of the nicest features to ease management of many social media platforms is the ability to “link” accounts and allow posts from Twitter to go to Facebook or LinkedIn posts to go to Twitter or Instagram to post on Facebook, but Google+ wishes to be the source of everything, which ends up limiting it if you like to have the ability to use any platform and have it cross post. I have a tool to help you with this and highly recommend using these to ease your day and fill in content around your own personal posts.

**Social Media Tools**

We all have day jobs and some companies even have social media sites blocked internally. This doesn’t mean that you can’t be involved in personal branding. There are a number of tools that can ease the demand and even allow you to post automatically while you are at work.

I work from home, have little to no coworker interaction throughout the day and yet I still spend very little time actually interacting on social media platforms. I post content and previous blogs 24X7 and I’ve often been asked how I manage this. Automating this is done through a number of tools that allow me to schedule out my posts when it’s convenient for me, which is crucial considering the demands on my time between work, volunteer work for user groups and a busy mother to three teens.

The web site http://buffer.com lets a user set up an account, add social media accounts to post to, even install an add-on to their web browser to easily “buffer” stories out to the social media accounts you sign up with your buffer account. Content is provided in your feed to be pushed out as posts by a formula set up of your interests. If a story only provides value to one of the accounts that are signed up, you can “uncheck” the ones that you wish not to post to and send the story out only to the social media platforms that will benefit. Lastly, you can also add Google+ to buffer, which doesn’t play well with social media platforms like Twitter, Facebook, Instagram and others that you can set up to auto-post to each other. This buffer slowly empties on the interval you’ve set up and allows you to provide content without requiring constant website content searching, tweeting and posting.

As I mentioned http://klout.com earlier, the site offers a similar service. Not only are you provided a score of your social media value, but you can also set up schedules to push information about your interests which will feed content to the accounts that are part of your klout score. Unlike buffer, that for free, can only “buffer” so much content before asking to upgrade to a paid subscription, klout allows you to schedule as much as you want, as far out as you choose.

The next opportunity for automation of social media content is for bloggers. If you have a good amount of blog content, you can set up a plug-in that will “retweet” old blog posts randomly or from categories on a regular basis to Twitter. Many of these allow you to set up a schedule of how often, (mine are on a once every 24hr interval) and it’s helpful to add a clear tweet indicator that the post is being retweeted, (I use “oldie but goodie” to identify this type of content.) You’ll be surprised how valuable old content can be to personal branding to those that follow you. No matter where you host your blog, (blogspot, wordpress, etc.) there should be a plugin that can be set up to retweet old posts, recycling at its best.

**Branding**

In a world of everyone wanting to be heard in a chaos of content, it’s important to get your brand out there. There are some that choose to do this with a reality TV type of drama, but that 15 minutes of fame can come with a great cost. If you want followers who are interested in you technically and for the topics you feel passionately about, choose to post on what is important to you and:

Remember that nothing disappears from the internet- choose your words wisely.

If a follower is providing nothing but negative comments and no value to you or your fellow coworkers, don’t hesitate to block them. This is extremely important to women who are building their personal brand. There is very little protection from internet “stalkers” and there is no reasoning with them. Blocking them is your best avenue to peace.

Only provide information about your personal life that you are comfortable with. Your brand should be about you, but what you feel offers value to the brand.

99% of negatives can be turned into a positive if approached from the right angle. A good example, women are often judged by their appearance and this can be difficult online. I’ve chosen to focus on my “goth image” in the work place. This is unexpected for first appearances and helps to deter many assumptions or stereotypes, allowing me to side-step common categorization.

What you think isn’t interesting about your topic might be very interesting to your audience. Blog, post and write on numerous topics and don’t be surprised if your most popular topic isn’t what you would have guessed.

The topic of personal branding is incredibly complex and there are a ton of tools out there built to support the marketing of companies all the way down to advertising individuals. As a technical professional, building your brand is all about you and having a few tips and tricks to simplify the process.

# SQL>UPDATE • Fall 2014 19
NoSQL is a “disruptive innovation” in the sense used by Harvard professor Clayton M. Christensen. In The Innovator’s Dilemma: When New Technologies Cause Great Firms to Fail, Professor Christensen defines disruptive innovations and explains why it is dangerous to ignore them:

“Generally, disruptive innovations were technologically straightforward, consisting of off-the-shelf components put together in a product architecture that was often simpler than prior approaches. They offered less of what customers in established markets wanted and so could rarely be initially employed there. They offered a different package of attributes valued only in emerging markets remote from, and unimportant to, the mainstream.”

The personal computer was an example of a disruptive innovation. It was initially targeted only at the home computing segment of the market. Established manufacturers of mainframe computers and minicomputers did not see PC technology as a threat to their bottom lines. Eventually, however, PC technology came to dominate the market and established computer manufacturers such as Digital Equipment Corporation, Prime, Wang, Nixdorf, Apollo, and Silicon Graphics went out of business.

So where lies the dilemma? Professor Christensen explains: “In every company, every day, every year, people are going into senior management, knocking on the door saying: ‘I got a new product for us.’ And some of those entail making better products that you can sell for higher prices to your best customers. A disruptive innovation generally has to cause you to go after new markets, people who aren’t your customers. And the product that you want to sell them is something that is just so much more affordable and simple that your current customers can’t buy it. And so the choice that you have to make is: Should we make better products that we can sell for better profits to our best customers. Or maybe we ought to make worse products that none of our customers would buy that would ruin our margins. What should we do? And that really is the dilemma.”

Exactly in the manner that Christensen described, the e-commerce pioneer Amazon created an in-house product called Dynamo in 2007 to meet the performance, scalability, and availability needs of its own e-commerce platform after it concluded that mainstream database management systems were not capable of satisfying those needs. The most notable aspect of Dynamo was the apparent break with the relational model; there was no mention of relations, relational algebra, or SQL.

Dynamo Requirements and Assumptions

Amazon started out by using the Oracle Database for its e-commerce platform, but later switched to a proprietary database management system called Dynamo that it built in-house. Dynamo is the archetypal NoSQL product; it embodies all the innovations of the NoSQL camp. The Dynamo requirements and assumptions are documented in the paper Dynamo: Amazon’s Highly Available Key-value Store published in 2007. Here are some excerpts from that paper:

“Customers should be able to view and add items to their shopping cart even if disks are failing, network routes are flapping, or data centers are being destroyed by tornados. Therefore, the service responsible for managing shopping carts requires that it can always write to and read from its data store, and that its data needs to be available across multiple data centers.”

“There are many services on Amazon’s platform that only need primary-key access to a data store. For many services, such as those that provide best seller lists, shopping carts, customer preferences, session management, sales rank, and product catalog, the common pattern of using a relational database would lead to inefficiencies and limit scale and availability. Dynamo provides a simple primary-key only interface to meet the requirements of these applications.”

“Experience at Amazon has shown that data stores that provide ACID guarantees tend to have poor availability.”

“Dynamo targets applications that operate with weaker consistency (the “C” in ACID) if this results in high availability.”

“…since each service uses its distinct instance of Dynamo, its initial design targets a scale of up to hundreds of storage hosts.”

To paraphrase, Amazon’s requirements were extreme performance, extreme scalability, and extreme availability.

Functional Segmentation

Amazon’s pivotal design decision was to break its monolithic enterprise-wide database service into simpler component services such as a best-seller list service, a shopping cart service, a customer preferences service, a sales rank service, and a product catalog service. This avoided a single point of failure. In an interview for the NoCOUG Journal, Amazon’s first database administrator, Jeremiah Wilton explains the rationale behind Amazon’s approach:
As an example, the shopping cart service should not be affected if the checkout service is unavailable or not performing well. I said that this was the pivotal design decision made by Amazon. I cannot emphasize this enough. If you resist functional segmentation, you are not ready for NoSQL. If you miss the point, you will not understand NoSQL.

Note that functional segmentation results in simple hierarchical schemas. Here is an example of a simple hierarchical schema from Ted Codd’s 1970 paper on the relational model. This simple schema stores information about employees, their children, their job histories, and their salary histories.

employee (man#, name, birthdate)
children (man#, childname, birthyear)
jobhistory (man#, jobdate, title)
salaryhistory (man#, jobdate, salarydate, salary)

Functional segmentation is the underpinning of NoSQL technology, but it does not present a conflict with the relational model; it is simply a physical database design decision. Each functional segment is usually assigned its own standalone database. The collection of functional segments could be regarded as a single distributed database. However, distributed transactions are forbidden in the NoSQL world. Functional segmentation can therefore result in temporary inconsistencies if, for example, the shopping cart data is not in the same database as the product catalog and occasional inconsistencies result. As an Amazon customer, I occasionally leave items in my shopping cart but don’t complete a purchase. When I resume shopping, I sometimes get a notification that an item in my shopping cart is no longer in stock or has been repriced. This technique is called “eventual consistency.”

“At eBay, we allow absolutely no client-side or distributed transactions of any kind – no two-phase commit. In certain well-defined situations, we will combine multiple statements on a single database into a single transactional operation. For the most part, however, individual statements are auto-committed. While this intentional relaxation of orthodox ACID properties does not guarantee immediate consistency everywhere, the reality is that most systems are available the vast majority of the time. Of course, we do employ various techniques to help the system reach eventual consistency: careful ordering of database operations, asynchronous recovery events, and reconciliation or settlement batches. We choose the technique according to the consistency demands of the particular use case.” (Scalability Best Practices: Lessons from eBay)

Sharding

Amazon’s next design decision was “sharding” or horizontal partitioning of all the tables in a hierarchical schema. Hash-partitioning is typically used. Each table is partitioned in the same way as the other tables in the schema and each set of partitions are placed in a separate database referred to as a “shard.” The shards are independent of each other; that is, there is no clustering as in Oracle RAC.

Note that the hierarchical schemas that result from functional segmentation are always shardable; that is, hierarchical schemas are shardable by definition.

Returning to the example from Ted Codd’s 1970 paper on the relational model:

employee (man#, name, birthdate) with primary key (man#)
children (man#, childname, birthyear) with primary key (man#, childname)
jobhistory (man#, jobdate, title) with primary key (man#, jobdate)
salaryhistory (man#, jobdate, salarydate, salary) with primary key (man#, jobdate, salarydate)

Note that the jobhistory, salaryhistory, and children tables have composite keys. In each case, the leading column of the composite key is the man#. Therefore, all four tables can be partitioned using the man#.

Sharding is an essential component of NoSQL designs but it does not present a conflict with the relational model; it too is simply a physical database design decision. In the relational model, the collection of standalone databases or shards can be logically viewed as a single distributed database.

Replication and Eventual Consistency

The Dynamo developers saw that one of the keys to extreme availability was data replication. Multiple copies of the shopping cart are allowed to exist and, if one of the replicas becomes unresponsive, the data can be served by one of the other replicas. However, because of network latencies, the copies may occasionally get out of sync and the customer may occasionally encounter a stale version of the shopping cart. Once again, this can be handled appropriately by the application tier; the node that falls behind can catch up eventually or inconsistencies can be detected and resolved at an opportune time, such as at checkout. This technique is called “eventual consistency.”

The inventor of relational theory, Dr. Codd, was acutely aware of the potential overhead of consistency checking. In his 1970 paper, he said:

“There are, of course, several possible ways in which a system can detect inconsistencies and respond to them. In one approach the system checks for possible inconsistency whenever an insertion, deletion, or key update occurs. Naturally, such checking will slow these operations down. [emphasis added] If an inconsistency has been generated, details are logged internally, and if it is not remedied within some reasonable time interval, either the
user or someone responsible for the security and integrity of the data is notified. Another approach is to conduct consistency checking as a batch operation once a day or less frequently.”

In other words, the inventor of relational theory would not have found a conflict between his relational model and the “eventual consistency” that is one of the hallmarks of the NoSQL products of today. However, the Dynamo developers imagined a conflict because it quite understandably conflated the relational model with the ACID guarantees of database management systems. However, ACID has nothing to do with the relational model per se (although relational theory does come in very handy in defining consistency constraints); pre-relational database management systems such as IMS provided ACID guarantees and so did post-relational object-oriented database management systems.

The tradeoff between consistency and performance is as important in the wired world of today as it was in Dr. Codd’s world. Synchronous replication is rarely used in the relational camp so we cannot frown at Dynamo for not using it. Application developers in the relational camp are warned about the negative impact of consistency checking so we cannot frown on Dynamo’s decision to permit temporary inconsistencies between functional segments.

• “Using primary and foreign keys can impact performance. Avoid using them when possible.” (http://docs.oracle.com/cd/E17904_01/core.1111/e10108/adapters.htm#BABCCCHH)

• “For performance reasons, the Oracle BPEL Process Manager, Oracle Mediator, human workflow, Oracle B2B, SOA Infrastructure, and Oracle BPM Suite schemas have no foreign key constraints to enforce integrity.” (http://docs.oracle.com/cd/E23943_01/admin.1111/e10226/soaadm_in_partition.htm#CJCJIII)

• “For database independence, applications typically do not store the primary key-foreign key relationships in the database itself; rather, the relationships are enforced in the application.” (http://docs.oracle.com/cd/E25178_01/fusion-apps.1111/e14496/securing.htm#CHDDGFHH)

• “The ETL process commonly verifies that certain constraints are true. For example, it can validate all of the foreign keys in the data coming into the fact table. This means that you can trust it to provide clean data, instead of implementing constraints in the data warehouse.” (http://docs.oracle.com/cd/E24693_01/server.1120/e16579/constra.htm#1006300)

The False Premise of NoSQL

The final hurdle was extreme performance, and that’s where the Dynamo developers went astray. The Dynamo developers believed that the relational model imposes a “join penalty” and therefore chose to store data as “blobs.” This objection to the relational model is colorfully summarized by the following statement attributed to Esther Dyson, the editor of the Release 1.0 newsletter,

“Using tables to store objects is like driving your car home and then disassembling it to put it in the garage. It can be assembled again in the morning, but one eventually asks whether this is the most efficient way to park a car.”

The statement dates back to 1988 and was much quoted when object-oriented databases were in vogue.

Since the shopping cart is an object, doesn’t disassembling it for storage make subsequent data retrieval and updates inefficient? The belief stems from an unfounded assumption that has found its way into every relational DBMS—that every table should map to physical storage. In reality, the relational model is a logical model and, therefore, it does not concern itself with storage details at all. It would be perfectly legitimate to store the shopping cart in a physical form that resembled a shopping cart while still offering a relational model of the data complete with SQL. In other words, the physical representation could be optimized for the most important use case—retrieving the entire shopping-cart object using its key—without affecting the relational model of the data. It would also be perfectly legitimate to provide a non-relational API for the important use cases. Dr. Codd himself gave conditional blessing to such non-relational APIs in his 1985 Computerworld article, “Is Your DBMS Really Relational?”, in which he says,

“If a relational system has a low-level (single-record-at-a-time) language, that low level [should not] be used to subvert or bypass the integrity rules and constraints expressed in the higher level relational language (multiple-records-at-a-time).”

The key-blob or “key-value” approach used by Dynamo and successor products would be called “zeroth” normal form in relational terminology. In his 1970 paper, Dr. Codd says:

“Nonatomic values can be discussed within the relational framework. Thus, some domains may have relations as elements. These relations may, in turn, be defined on nonsimple domains, and so on. For example, one of the domains on which the relation employee is defined might be salary history. An element of the salary history domain is a binary relation defined on the domain date and the domain salary. The salary history domain is the set of all such binary relations. At any instant of time there are as many instances of the salary history relation in the data bank as there are employees. In contrast, there is only one instance of the employee relation.”

In common parlance, a relation with non-simple domains is said to be in “zeroth” normal form or unnormalized. Dr. Codd suggested that unnormalized relations should be normalized for ease of use. Here again is the unnormalized employee relation from Dr. Codd’s paper:

```
employee (employee#, name, birthdate, jobhistory (jobdate, title, salaryhistory (salarydate, salary)), children (childname, birthyear))
```

The above unnormalized relation can be decomposed into four normalized relations as follows:

```
employee’ (employee#, name, birthdate)
jobhistory’ (employee#, jobdate, title)
salaryhistory’ (employee#, jobdate, salarydate, salary)
children’ (employee#, childname, birthyear)
```

However, this is not to suggest that these normalized relations must necessarily be mapped to individual buckets of physical stor-
age. Dr. Codd differentiated between the stored set, the named set, and the expressible set. In the above example, we have one unnormalized relation and four normalized relations, if we preferred it, the unnormalized employee relation could be the only member of the stored set. Alternatively, if we preferred it, all five relations could be part of the stored set; that is, we could legitimately store redundant representations of the data. However, the common belief blessed by current practice is that the normalized relations should be the only members of the stored set.

Even if the stored set contains only normalized relations, they need not map to different buckets of physical storage. Oracle Database is unique among mainstream database management systems in providing a convenient construct called the “table cluster” that is suitable for hierarchical schemas. In Dr. Codd’s example, employee# would be the cluster key, and rows corresponding to the same cluster key from all four tables could be stored in the same physical block on disk thus avoiding the join penalty. If the cluster was a “hash cluster,” no indexes would be required for the use case of retrieving records belonging to a single cluster key. A demonstration is available at is available at [http://iggyfernandez.wordpress.com/2013/12/30/the-twelve-days-of-nosql/day-six-the-false-premise-of-nosql/](http://iggyfernandez.wordpress.com/2013/12/30/the-twelve-days-of-nosql/day-six-the-false-premise-of-nosql/).

**Schemaless Design**

The final innovation of the NoSQL camp is “schemaless design.” In database management systems of the NoSQL kind, data is stored in “blobs” and documents. The database management system does not police their structure. In mainstream database management systems on the other hand, doctrinal purity requires that the schema be designed before data is inserted. Let’s do a thought experiment.

Suppose that we don’t have a schema and let’s suppose that the following facts are known.

- Iggy Fernandez is an employee with EMPLOYEE_ID=1 and SALARY=$1000.
- Mogens Norgaard is a commissioned employee with EMPLOYEE_ID=2, SALARY=€1000, and COMMISSION_PCT=25.
- Morten Egan is a commissioned employee with EMPLOYEE_ID=3, SALARY=€1000, and unknown COMMISSION_PCT.

Could we ask the following questions and expect to receive correct answers?

- **Question**: What is the salary of Iggy Fernandez?
  - Expected answer: $1000.
- **Question**: What is the commission percentage of Iggy Fernandez?
  - Expected answer: Invalid question.
- **Question**: What is the commission percentage of Mogens Norgaard?
  - Expected answer: 25%
- **Question**: What is the commission percentage of Morten Egan?
  - Expected answer: Unknown.

If we humans can process the above data and correctly answer the above questions, then surely we can program computers to do so.

The above data could be modeled with the following three relations. It is certainly disruptive to suggest that this be done on the fly by the database management system but not outside the realm of possibility.

```sql
CREATE TABLE EMPLOYEES
(EMPLOYEE_ID NOT NULL NUMBER(6)
EMPLOYEE_NAME VARCHAR2(128)
)
CREATE TABLE UNCOMMISSIONED_EMPLOYEES
(EMPLOYEE_ID NOT NULL NUMBER(6)
SALARY NUMBER(8,2)
)
CREATE TABLE COMMISSIONED_EMPLOYEES
(EMPLOYEE_ID NOT NULL NUMBER(6)
SALARY NUMBER(8,2)
COMMISSION_PCT NUMBER(2,2)
)
```

A NoSQL company called Hadapt has already stepped forward with such a feature:

*While it is true that SQL requires a schema, it is entirely untrue that the user has to define this schema in advance before query processing. There are many data sets out there, including JSON, XML, and generic key-value data sets that are self-describing — each value is associated with some key that describes what entity attribute this value is associated with [emphasis added]. If these data sets are stored in Hadoop, there is no reason why Hadoop cannot automatically generate a virtual schema against which SQL queries can be issued. And if this is true, users should not be forced to define a schema before using a SQL-on-Hadoop solution they should be able to effortlessly issue SQL against a schema that was automatically generated for them when data was loaded into Hadoop.*

*Thanks to the hard work of many people at Hadapt from several different groups, including the science team who developed an initial design of the feature, the engineering team who continued to refine the design and integrate it into Hadapt’s SQL-on-Hadoop solution, and the customer solutions team who worked with early customers to test and collect feedback on the functionality of this feature, this feature is now available in Hadoop.* ([http://hadapt.com/blog/2013/10/28/all-sql-on-hadoop-solutions-are-missing-the-point-of-hadoop/](http://hadapt.com/blog/2013/10/28/all-sql-on-hadoop-solutions-are-missing-the-point-of-hadoop/))

This is not really new ground. Oracle Database provides the ability to convert XML documents into relational tables ([http://docs.oracle.com/cd/E11882_01/appdev.112/e23094/xdb01int.htm#ADXDB0120](http://docs.oracle.com/cd/E11882_01/appdev.112/e23094/xdb01int.htm#ADXDB0120)) though it ought to be possible to view XML data as tables while physically storing it in XML format in order to benefit certain use cases. It should also be possible to redundantly store data in both XML and relational formats in order to benefit other use cases.

In “Extending the Database Relational Model to Capture More Meaning,” Dr. Codd explains how a “formatted database” is created from a collection of facts:

*“Suppose we think of a database initially as a set of formulas in first-order predicate logic. Further, each formula has no free variables and is in as atomic a form as possible (e.g., A & B would be replaced by the component formulas A, B). Now suppose that most of the formulas are simple assertions of the form Pab...z (where...continued on page 26*
A
fter working for a variety of companies in the 1980s, after working for Oracle in the 1990s, after trying (and failing) to build a company with friends at the turn of the century, and after more than a decade working as an independent consultant in this new century, I found myself in a professional dilemma last year.

I know I need to work at least another fifteen years to be able to retire. I had survived the nastiest economic downturn since the Great Depression while self-employed, and felt ready to take on the economic upswing, so I was confident that I could work steadily for the next 15 years or more.

Problem was: I was getting bored.

I loved my work. I enjoy the “sleuthiness” and the forensic challenge of finding performance problems, testing and recommending solutions, and finding a way to describe it clearly so that my customer can make the right decision. I am confident that I can identify and address any database performance problem facing an application built on Oracle database, and I’ve had dozens of successful consulting engagements to bear witness. I have a legion of happy customers and a seemingly endless supply of referrals.

Being confident is a great feeling, and I had spent the past several years just enjoying that feeling of confidence on each engagement, relishing the challenge, the chase, and the conclusion.

But it was becoming routine. The best explanation I have is that I felt like a hammer, and I addressed every problem as if it was some form of nail. I could feel my mental acuity ossifying.

Then, opportunity knocked, in an unexpected form from an unexpected direction.

I have a friend and colleague whom I’ve known for almost 20 years, named Kyle Hailey. Kyle is one of those notably brilliant people, the kind of person to whom you pay attention immediately, whether you meet online or in person. We had both worked at Oracle in the 1990s, and I had stayed in touch with him over the years since.

About four years ago, I became aware that he was involved in a new venture, a startup company called Delphix. I wasn’t sure what it was about, but I paid attention because Kyle was involved. Then, about 2 years ago, I was working as a DBA for a local Colorado company who had decided to evaluate Delphix’s product. Representing my customer, my job was to prevent a disaster from taking place. I was to determine if the product had any merit, if the problems being experienced were insurmountable, and if so let my customer know so they could kill the project.

What does Delphix do? The product itself is a storage appliance on a virtual machine in the data center. It uses sophisticated compression, deduplication, and copy-on-write technical to clone production databases for non-production usage such as development and testing. It does this by importing a copy of the production database into the application, compressing that base copy down to 25-30% of its original size. Then, it provides “virtual databases” from that base copy, each virtual database consuming almost no space at first, since almost everything is read from the base copy. As changes are made to each virtual database, copy-on-write technology stores only those changes for only that virtual database. So, each virtual database is presented as a full image of the source database, but costs practically nothing. Even though the Oracle database instances reside on separate servers, the virtual database actually resides on the Delphix engine appliance and is presented via NFS.

I was asked to understand why the virtual databases were slow.

On the other end of the phone was Kyle, and he was easily able to show me with repeatable tests where and what the nature of the performance problems were, and that they were predictable and resolvable.

In the end, I conveyed this information back to my customer, and didn’t learn until months later, after my consulting engagement had completed, that they had in fact purchased the Delphix product.

But I’m not really writing just about Delphix’s product, even though it is very cool and quite earthshaking. Rather, I’m writing about something bigger that has stormed into our industry, bringing to fruition something that I had tried -- and failed -- to accomplish at the turn of the century. Back then, at the turn of the century, when some colleagues and I tried to build a hosted-application services company, we failed in two ways: 1) we were ahead of our time and 2) we chose the wrong customers.

Being ahead of one’s time is not a failure, strictly speaking. It shows a clarity of vision, but bad timing.

However, choosing the wrong customers is absolutely a failure.

Before I explain, please be aware that I’m going to change most of the names, to protect any innocent bystanders...
After leaving Oracle in July 1998, I founded my own little company-of-one, called Evergreen Database Technologies. I had had enough of corporate life, the dot-com boom was booming, and I wanted to operate as an independent consultant. I had been living in Evergreen in the foothills above Denver for years, so the choice of name for my company was not a brilliant leap of imagination. Business was brisk, even a bit overheated as Y2K approached, and I was happy.

In early 2000, I had been working with another young company called Upstart, and we felt that the information technology (IT) industry was heading in one inescapable direction: hosted services. So I joined Upstart and we decided that providing hosted Oracle E-Business Suites (EBS) was a good start. EBS is employed in hundreds of different industries and is infinitely customizable, so in order to avoid being eaten alive by customization requests by our potential customers, we at Upstart would have to focus on a specific industry and pre-customize EBS according to the best practices for that industry. We chose the telecommunications industry, because it was an even bigger industry in Colorado then as it is now. We wanted to focus on small customers, being small ourselves. At that time, small telecommunications companies were plentiful because of governmental deregulation in the industry. These companies offered DSL internet and phone services, and in June 2000 our market research told us it was a US$9 billion industry segment and growing.

Unfortunately, all of the big primary telecommunications carriers were also watching and were quick to catch on, and they entered the DSL market themselves, provided cheaper and better service, and put the small companies out of business practically overnight. “Overnight” is not a big exaggeration.

By October 2000, we at Upstart were stunned to discover that our customers had literally ceased answering their phones, which is bitterly ironic and chillingly ominous in the case of telecommunications companies, well as the companies who had hoped to make money from those companies, such as Upstart. We at Upstart had a difference of opinion on how to proceed, with some of the principals arguing to continue sucking down venture-capital funding and stay the course, while others argued that we had to find some way, any way, to resume generating revenue on which to survive. I stood in the latter category, and advocated returning to Upstart’s previous business model of consulting services, but the principals who wanted to stay the course with the hosting model wouldn’t budge. By December 2000, Upstart was bankrupt, and I jumped back into my own personal lifeboat, my little consulting services company-of-one, Evergreen Database Technologies.

This was a scenario that I’m told was repeated in many companies that flamed out during the “dot-bomb” era. There are a million stories in the big city, and mine is one of them.

But let’s just suppose that Upstart had chosen a different customer base, one that didn’t disappear completely within months? Would it still have survived?

There is a good chance that we would still have failed due to being ahead of our times. Hosted applications, which today are called “cloud services”, were made more difficult back then by the fact that EBS software was configured with the intention of occupying entire servers. The documentation from Oracle assumed so, and support at Oracle assumed the same. This meant that we, Upstart, had to provision several entire servers for each customer, which is precisely what they were doing for themselves. As a result, we could not operate any more cheaply than our customers had, resulting in no cost savings or efficiencies in that area, eating into our profit margins.

What we needed at the time was server virtualization, which came along a few years later in the form of companies like VMware. Not until virtualization permitted us to run enterprise software on virtual machines, which could be stacked en masse on physical machines, could we have operated in a manner to save costs and make money.

Fast forward to today.

Today, server virtualization is everywhere. You can create virtual machines on your laptop, emulating almost any operating system that has ever existed, and creating them in such a way that finally makes full use of all the resources of real, physical servers. No longer would system administrators rack, wire, and network physical servers using customized specifications for each server. Instead, virtual machines could be configured according to the customized specifications, and those virtual machines run by the dozens on physical machines.

The advent of virtual machines brought about the operations paradise of abstracting computer servers completely into software, so that they could be built, configured, operated, and destroyed entirely like the software constructs they were. No more racking and wiring, one server per application. Now, banks of “blades” were racked and wired generically, and virtual machines balanced within and across blades, with busier virtual machines moving toward available CPU, memory, and networking resources and quieter virtual machines yielding CPU, memory, and networking to others.

Everything was virtualized, and all was good.

Except storage.

Think about it. It is easy and cheap to provision another virtual machine, using fractions of CPU cores and RAM. But each of those virtual machines needs a full image of operating system, application software, and database. While server virtualization permitted data centers to use physical servers more efficiently, it caused a positive supernova explosion in storage. So much so that analysts like Gartner have predicted a
NoSQL Taxonomy

NoSQL databases can be classified into the following categories:

• **Key-value stores:** The archetype is Amazon Dynamo of which DynamoDB is the commercial successor. Key-value stores basically allow applications to "put" and "get" values but each product has differentiators. For example, DynamoDB supports "tables" (namespaces) while Oracle NoSQL Database offers "major" and "minor" key paths.

• **Document stores:** While key-value stores view values as uninterpreted strings, document stores allow values to be managed using formats such as JSON (JavaScript Object Notation) which are conceptually similar to XML. This allows key-value pairs to be indexed by any component of the value just as XML data can be indexed in mainstream database management systems.

• **Column-family stores:** Column-family stores allow data associated with a single key to be spread over multiple storage nodes. Each storage node only stores a subset of the data associated with the key; hence the name "column-family." A key is therefore composed of a "row key" and a "column key."

• **Graph databases:** Graph databases are non-relational databases that use graph concepts such as nodes and edges to solve certain classes of problems: for example, the shortest route between two towns on a map. The concepts of functional segmentation, sharding, replication, eventual consistency, and schemaless design do not apply to graph databases so I will not discuss graph databases.

NoSQL Buyer’s Guide

NoSQL products are numerous and rapidly evolving. There is a crying need for a continuously updated encyclopedia of NoSQL products but none exists. There is a crying need for an independent benchmarking organization but none exists. My best advice is to do a proof of concept (POC) as well as a PSR (Performance, Scalability, and Reliability) test before committing to using a NoSQL product. Back in 1985, Dr. Codd had words of advice for those who were debating between the new relational products and the established pre-relational products of the time.

"Any buyer confronted with the decision of which DBMS to acquire should weigh three factors heavily.

The first factor is the buyer's performance requirements, often expressed in terms of the number of transactions that must be executed per second. The average complexity of each transaction is also an important consideration.

Only if the performance requirements are extremely severe should buyers rule out present relational DBMS products on this basis. Even then buyers should design performance tests of their own, rather than rely on vendor-designed tests or vendor-declared strategies. [emphasis added]

The second factor is reduced costs for developing new databases and new application programs …

The third factor is protecting future investments in application programs by acquiring a DBMS with a solid theoretical foundation …

In every case, a relational DBMS wins on factors two and three. In many cases, it can win on factor one also—in spite of all the myths about performance."

—An Evaluation Scheme for Database Management Systems

The above advice is as solid today as it was in Dr. Codd’s day.

Oracle NoSQL Database

In May 2011, Oracle Corporation published a white paper titled "Debunking the NoSQL Hype," the last words being “Go for the tried and true path. Don’t be risking your data on NoSQL databases.” However, in September of the same year, Oracle Corporation released Oracle NoSQL Database. Oracle suggested that the NoSQL approach was well-suited for certain applications:

“The Oracle NoSQL Database, with its ‘No Single Point of Failure’ architecture, is the right solution when data access is “simple” in nature and application demands exceed the volume or latency capability of traditional data management solutions. For example, click-stream data from high volume web sites, high-throughput event processing and social networking communications all represent application domains that produce extraordinary volumes of simple keyed data. Monitoring online retail behavior, accessing customer profiles, pulling up appropriate customer ads and storing and forwarding real-time communication are examples of domains requiring the ultimate in low-latency access. Highly distributed applications such as real-time sensor aggregation and scalable authentication also represent domains well-suited to Oracle NoSQL Database.‟—Oracle NoSQL Database

Oracle NoSQL Database has two features that distinguish it from other key-value stores: A key is the concatenation of a “major key path” and a “minor key path.” All records with the same “major key path” will be colocated on the same storage node. In addition, Oracle NoSQL provides transactional support for modifying multiple records with the same major key path.

Challenges to NoSQL

There are already proofs that performance, scalability, and reliability can be achieved without abandoning the relational model. For example, ScaleBase provides sharding and replication on top of MySQL storage nodes. Another good example to study is VoltDB which claims to be the world’s fastest OLTP database (though it has never published an audited TPC benchmark). A counter-example to Amazon is eBay which arguably has equal scale and equally high performance, scalability, and reliability require-
ments. eBay uses performance segmentation, sharding, replication, and eventual consistency but continues to use Oracle (and SQL) to manage local databases. I asked Randy Shoup, one of the architects of the eBay e-commerce platform, why eBay did not abandon Oracle Database and he answered in one word: “comfort.” Here are links to some of his presentations and articles on the eBay architecture:

- “eBay’s Scaling Odyssey: Growing and Evolving a Large eCommerce Site” (http://www.cs.cornell.edu/projects/ladis2008/materials/eBayScalingOdyssey_ShoupTravostino.pdf)
- The eBay Architecture: Striking a balance between site stability, feature velocity, performance, and cost (http://www.infoq.com/interviews/shoup-ebay-architecture)
- The eBay Architecture: Striking a balance between site stability, feature velocity, performance, and cost (http://www.infoq.com/presentations/shoup-ebay-architectural-principles)
- The eBay Architecture: Striking a balance between site stability, feature velocity, performance, and cost (http://www.infoq.com/articles/ebay-scalability-best-practices)

The latest challenge to NoSQL comes from Google which recently created a new DBMS called F1 for its business-critical AdWords application. Google implemented a version of Oracle table clusters in order to avoid the join penalty. Here is a quote from Google’s paper:

“In recent years, conventional wisdom in the engineering community has been that if you need a highly scalable, high-throughput data store, the only viable option is to use a NoSQL key/value store, and to work around the lack of ACID transactional guarantees and the lack of conveniences like secondary indexes, SQL, and so on. When we sought a replacement for Google’s MySQL data store for the AdWords product, that option was simply not feasible: the complexity of dealing with a non-ACID data store in every part of our business logic would be too great, and there was simply no way our business could function without SQL queries. Instead of going NoSQL, we built F1, a distributed relational database system that combines high availability, the throughput and scalability of NoSQL systems, and the functionality, usability and consistency of traditional relational databases, including ACID transactions and SQL queries. Google’s core AdWords business is now running completely on F1. F1 provides the SQL database functionality that our developers are used to and our business requires. Unlike our MySQL solution, F1 is trivial to scale up by simply adding machines.”— F1: A Distributed SQL Database That Scales

**Summary**

The NoSQL camp put performance, scalability, and reliability front and center but lost the opportunity to take the relational model to the next level because it mistakenly believed that normalization dictates physical storage choices, that non-relational APIs are forbidden by the relational model, and that relational is synonymous with ACID (Atomicity, Consistency, Isolation, Durability).

The NoSQL camp created a number of innovations that are disruptive in the sense used by Harvard Business School professor Clayton Christensen: functional segmentation, sharding, replica-

tion, eventual consistency, and schemaless design. Since these innovations are compatible with the relational model, they should eventually be absorbed by mainstream database management systems.

Finally, I should point out that are very good reasons to criticize current NoSQL products; for example, lack of standards, primitive feature sets, primitive security, and primitive management tools, unproven claims, and traps for the unwary. MongoDB uses a database-wide lock for reads and writes. #nuffsaid.

Iggy has tons of Oracle Database experience and is the longtime editor of the NoCOUG Journal, the quarterly publication of the Northern California Oracle Users Group. He has a lot of opinions but is willing to change them when confronted with fresh facts. As the American philosopher Ralph Waldo Emerson said in 1841 in his essay on self-reliance,

“A foolish consistency is the hobgoblin of little minds, adored by little statesmen and philosophers and divines. … speak what you think to-day in words as hard as cannon-balls, and tomorrow speak what to-morrow thinks in hard words again, though it contradict everything you said to-day.”

---

Tim Gorman.....continued from page 25

“data crisis” before the end of the decade.

This is where Delphix comes in.

By virtualizing data as well as servers, it is now truly fast, cheap, and easy to provision entire virtual environments. Delphix works with Oracle, and it also works with SQL Server, PostgreSQL, and Sybase. Even more importantly, it also virtualizes file-systems, so that application software as well as databases can be virtualized.

So earlier this year, Kyle contacted me and asked if I would be interested in joining Delphix. My first reaction was the one I always had, which is “no thanks, I’ve already got a job”. Business was still brisk, I never had downtime, and the economy was improving. I had successfully been operating as an independent consultant for most of the past 15 years. Why fix what wasn’t broken?

I wanted to try something new, to expand beyond what I had been doing for the past 25 years since I first joined Oracle. If it was a large established company beckoning, I wouldn’t have considered it for a second more. But a promising startup company, with a great idea and a four-year track record of success already, and still pre-IPO as well?

I couldn’t resist. What’s not to like?

It’s possible I’ve made an enormous mistake, but I don’t think so.

Not to turn excessively morbid, but all of us are just a heartbeat away from our common destination. I believe that, when I’m at the last moments of my life, the thing I fear will not be death itself, or pain, or leaving life behind.

It is regret that I fear.

And regret can take many forms, but the most painful regret will undoubtedly be what might have been, the opportunities passed or missed. Career is only one aspect of life, and I don’t want to give it too much significance. I’ve accumulated regrets in how I’ve lived my life, and how I’ve treated people in my life, and in some cases they are small but there are some which will always haunt me.

But with regards to my professional career, as Robert Frost said, I’ve taken the road less traveled, and that has made all the difference.
RMOUG Board of Directors

Meet Your Board

John Peterson
President
E-mail: President@rmoug.org

Vince Giasolli
Vice President
Newsletter Director
E-mail: VicePresident@rmoug.org
NewsletterDir@rmoug.org

Chris Ostrowski
Secretary
E-mail: Secretary@rmoug.org

Ron Bich
Treasurer
E-mail: Treasurer@rmoug.org

Bobby Curtis
Social Media Director
E-mail: SocialMediaDir@rmoug.org

Andreas Katsarlis
Marketing Director
E-mail: MarketingDir@rmoug.org

Dan Hotka
Programs Director
E-mail: ProgramsDir@rmoug.org

Art Marshall
Web Director
E-mail: WebDir@rmoug.org

Chris Chase
SIGs Director
E-mail: SigsDir@rmoug.org

Bob Mason
Education Director
Scholarship Director
E-mail: EducationDir@rmoug.org
RMOUG Board of Directors

John Jeunnette
Training Days Technology
Vendor Support
Board Member Emeritus
E-mail: TrainingDaysDir@rmoug.org

Kumar Annamnidu
Board Member At Large
E-mail: MembersAtLarge@rmoug.org

Pat Van Buskirk
Board Member Emeritus
E-mail: BoardME@rmoug.org

Kellyn Pot’Vin
Training Days Director
Non-Voting Board Member
Emeritus
E-mail: TrainingDaysDir@rmoug.org

Al Gonzales
Board Member At Large
E-mail: MembersAtLarge@rmoug.org

Kellyn Pot’Vin
Training Days Director
Non-Voting Board Member
Emeritus
E-mail: TrainingDaysDir@rmoug.org

Mark James
Board Member At Large
E-mail: MembersAtLarge@rmoug.org

Heidi Kuhn
Executive Director
Voice Mail: (303) 948-1786
Fax: (303) 933-6603
E-mail: Admin@rmoug.org

Kathy Robb
Board Member Emeritus
Arisant, LLC
E-mail: BoardME@rmoug.org

Tim Gorman
Past President
E-mail: BoardME@rmoug.org

Peggy King
Board Member Emeritus
E-mail: BoardME@rmoug.org

Pat Van Buskirk
Board Member Emeritus
E-mail: BoardME@rmoug.org

HMS>UPDATE • Fall 2014 29
30

SQL>UPDATE • Fall 2014

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/06/2014</td>
<td>RMOUG</td>
<td>Deadline for RMOUG Training Days 2015 (TD2015) abstracts</td>
</tr>
<tr>
<td>10/08/2014</td>
<td>RMOUG</td>
<td>TD2015 abstract selection meeting</td>
</tr>
<tr>
<td>11/ 21/2014</td>
<td>RMOUG</td>
<td>FALL QEW at DTC, Englewood</td>
</tr>
<tr>
<td>11/15/2014</td>
<td>RMOUG</td>
<td>SQL&gt;Update article &amp; photo deadline</td>
</tr>
<tr>
<td>11/19/2014</td>
<td>NoCOUG</td>
<td>Autumn Conference</td>
</tr>
<tr>
<td>12/15/2014</td>
<td>RMOUG</td>
<td>SQL&gt;Update Publication Date</td>
</tr>
<tr>
<td>2/17-2/19/2015</td>
<td>RMOUG</td>
<td>Training Days, Denver (TD2015)</td>
</tr>
</tbody>
</table>

**Please note dates are subject to change. For the most current events calendar visit our website at [www.rmoug.org](http://www.rmoug.org).**

---

**SPONSOR A QUARTERLY EDUCATION WORKSHOP AND RECEIVE A HALF PRICE AD IN SQL>UPDATE**

<table>
<thead>
<tr>
<th>Breakfast</th>
<th>Discounted Ad Rate</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/4 Page</td>
<td>$350.00</td>
<td>$175.00</td>
</tr>
<tr>
<td>1/2 Page</td>
<td>$350.00</td>
<td>$312.50</td>
</tr>
<tr>
<td>Full Page</td>
<td>$350.00</td>
<td>$500.00</td>
</tr>
<tr>
<td>Inside Cover</td>
<td>$350.00</td>
<td>$625.00</td>
</tr>
<tr>
<td>Back Cover</td>
<td>$350.00</td>
<td>$750.00</td>
</tr>
</tbody>
</table>

**Join Us November 21, 2014 at the Oracle DTC Campus!**

**More Details [www.rmoug.org](http://www.rmoug.org)**

**Contact Dan Hotka - Programs Director - ProgramsDir@rmoug.org**

---

**Tell Us About Yourself**

Join us in sharing your Oracle experiences with other RMOUG members!

Tell us about your life, your job, or share your amusing Oracle anecdotes, tips and secrets!

Share your favorite photos with our members and have the chance to be published on our cover!

Prior to each issue, we ask members to present their favorite photo, regional to the Rocky Mountains and not previously published. Each issue has a theme - Spring, Summer, Fall and Winter - and we need a minimum format of 300 dpi at least 8.75” x 11.25”. So turn your camera vertically and imagine the words **RMOUG SQL>UPDATE** across the top!

Please submit all material to NewsletterDir@rmoug.org
Arisant ................................................................................................. 2
Confio ................................................................................................. 12
DBAK ................................................................................................. 2
Quarterly Education Workshop Sponsorship ............................... 31
SQL>Update Opportunities ............................................................... 30
Training Days 2015 ........................................................................... 32

Reach A Targeted Oracle Audience • Advertise Now!
A full page, full color ad in RMOUG SQL>UPDATE costs as little as
70 cents per printed magazine and even less for smaller ads.
Contact NewsletterDir@rmoug.org

Quarterly Education Workshops

November 21, 2014

Join us for our next Quarterly Education Workshop in November for presentations at
the Oracle DTC Campus. RMOUG hosts quarterly workshops in May, August and
November of each year with the fourth and largest educational event being Training
Days in February. Learn about the newest technologies, gain more insight into Oracle
techniques and enjoy the camaraderie of meeting with other Oracle professionals.

If you or your organization are interested in partnering with RMOUG to host an upcoming
meeting, or to submit an abstract for presentation, please contact
Dan Hotka, Programs Director at ProgramsDir@rmoug.org

Watch RMOUG’s Web Page for November Training Topics www.rmoug.org
Tracks
- Application Development
- Application Express and ADF
- Business Intelligence
- COAUG/Applications
- Database Administration
- Database Tools of the Trade
- Emergent (Big Data, Hadoop, Hive)
- Middleware
- Professional Empowerment

Highlights
- Two full days of training with a bonus half-day of two-hour deep-dive and hands-on sessions
- More than 100 technical sessions with new tracks
- Full exhibition
- ACE luncheon, WIT, and career panel
- Networking opportunities to meet and talk with Oracle ACE Directors and ACEs, product managers from Oracle, and other Oracle professionals

Abstract Submission Deadline: October 2, 2014

Benefits of Presenting:
- You will receive a complimentary conference registration.
- Presenting at a prestigious conference such as RMOUG Training Days looks good on your resume and demonstrates to your clients and/or employer that you are good at what you do.
- As you share your knowledge and experience with other Oracle professionals, you will also learn something new from your audience.
- Presenting at RMOUG is always a good experience.

Submit your abstract @ www.rmoug.org