CAN I BE A DBA TOO? MENTORING A NON-DBA

Siddha Upadhye – Network Solutions

Suhas Upadhye – Niwara LLC

IOUG – COLLABORATE08

1. INTRODUCTION

You have been administering Oracle and its many tools and technologies for the past 10-12 years and/or you have managed Oracle projects in the past wherein such wise DBA's have worked for you. You are now the Manager / Database Architect of a cutting edge system with all the requirement specifications already nailed down. All process diagrams, data-flow-diagrams, entity-relationship diagrams, etcetera have been now flushed out and also the web application has been coded, unit and system tested and thus is all ready to go. The project is going live in six weeks - BAM! You are going to lose your DBA and you need him to get the Lead Oracle Developer up to speed. What must the Database Administration/Architecture hand-over document say and what should the actual handover entail? Let us walk through such a requirement and equip ourselves with the necessary and sufficient tools to takeover a DBA's role.

2. WHAT IS COVERED

- Blueprint of a Complex Project
- Details on Environment Setup
- Oracle Database Set-up Parameters
- Database versioning
- Code Reviews
- Explain-Plans – document them
- Explain Plan Operations and Cost explained
- Oracle Meta-Data Tables
- Sample Database Architecture – A Birds Eye View
- Sample Database Archival Strategy
- Sample Handover Checklist
- Additional Aspects
- Summary

3. WHAT IS NOT COVERED

The topics herein are great topics in their own right and would merit their own separate sessions.

- Qualities that are necessary to be a DBA
- Qualities of a DBA-Mentor
- A 60-minute crash course on 'How to be a DBA'
- A 60-minute crash course on 'How to be a DBA-mentor'
- Explanation of Oracle Internals

4. BLUEPRINT OF A COMPLEX PROJECT

Here is a fictional case-study provided to explain the variety of challenges that may be involved in a complex project. An organization, in accordance with their Business and Technology Plan, may seek (for example) to replace its existing legacy systems/PC-based applications, and manual processes with current technology. It is expected that the implementation of new information systems will provide significant benefits to the customers, organization staff, and other organization and federal agencies that interact with the organization.

The key automation objectives and standards of this project could include:

- Implementing a relational database management system that integrates the organizations inventory, order-processing and financial information. The organization has considerable experience with Oracle. Accordingly, the organization would prefer (but does not mandate) that the new system be developed in the Oracle environment. If the respondent is proposing a system developed in a database environment other than Oracle, the proposal should include a detailed analysis of the benefits the organization can expect by operating in the proposed environment.

- Running new systems in an open, industry-standard hardware platform and operating environment. The organization has considerable experience with the IBM AIX and Sun Solaris operating environments. Accordingly, the organization would prefer (but does not mandate) that the new system runs under a version of UNIX. If the respondent is proposing a system that does not run under UNIX, the proposal should include a detailed analysis of the benefits the organization can expect by operating in the proposed environment.

- Running new systems in a client-server environment over the organization's Wide Area Network (WAN). The organization must provide on-line system access to N branch offices and a number of other organization sites over the existing WAN. Accordingly, the proposed system must support TCP/IP and Ethernet protocols.

- Implementing applications written with current development tools that supports a browser-based interface or a Graphical User Interface (GUI). The organization would prefer (but does not mandate) a browser-based interface to facilitate local and remote connectivity to the new system.

- Improving system performance. The organization seeks sub-second response time for certain index field lookups and transaction posting processes and seeks to minimize turnaround time for end-to-end transaction processing of vital state records. Respondents are required to propose the hardware and telecommunications infrastructure needed to accomplish this objective.

- Improving system reliability. The organization expects the new system to be available 24 hours a day, 7 days a week. The organization expects no more than 1% downtime of its critical business applications. Respondents must propose fault-tolerant hardware and a configuration that provides the organization with redundancy so that this objective can be accomplished.

- Maintaining the flexibility to interface with other organization and national systems, messaging systems, and information clearing houses.

- Providing additional opportunities for customer self-service through Internet-based technology.

- Providing flexible reporting and statistical analysis tools to users and organization management.

5. DETAILS ON ENVIRONMENT SETUP

Given the complexities of the sample project as described above, let us now work our way to find out and get accustomed as to what is involved in the Environment Setup for such a project. You can perhaps identify with

one or more of these in the projects you have worked so far – these details are however rarely or never noticed by a Lead Developer, who as you know is going to fill in the (big) shoes of the soon-to-be "ex-DBA".

5.1. DATABASE DESIGN

Description & Documentation of Processes

1. Document the implementation of database changes and code are migrated through the development model, and on what schedule.
2. Create Full DB release documentation to include database creation scripts (DDL) for tables, constraints, indexes, triggers, packages, procedures, functions, views, etc., deployment practices to each environment, etc. Include a detailed description.
3. Database Model Change Report - Detail of each modification & reason for modification.
4. Document database release process.
5. Document participation in design reviews process.
6. Complete the documentation to correctly describe all existing data dictionary elements.
7. Document strategy for final implementation of indexing, unique keys, referential integrity keys, etc. to the database at the latest release as it relates to tuning the database.
8. Incorporate all NT Share System Materials into the Configuration Manager Tool Repository
9. Ensure all ClearCase & ClearQuest Repository information is appropriately located and 'Repository Directory' listing is updated to include the incorporation of new entries.

Update Design Documents

1. Data Model – Logical ERD through latest database model release
2. Data Model – Physical Design through latest database model release

Document Open Design Issues

1. Provide a detailed list of any outstanding database design issues (include deadlocks, bound variables not used consistently, database normalization,. resource contention, performance issues, etc.) and recommend possible resolutions
2. Document issues related to final implementation of indexing, unique keys, referential integrity keys, etc. to the database at the latest release.

Provide Documentation to Transition DBA

1. Provide electronic updateable copies of each design document and all scripts or source code. The electronic version will be transferred to the Transition Manager within the appropriate ClearCase and ClearQuest Repository area. Ensure design documents, scripts, and source code is checked into ClearCase & ClearQuest Repository in the designated location as outlined in the 'Repository Directory'.

5.2. DATABASE ADMINISTRATION

Description & Documentation of Processes

1. Document database configuration information.
2. Document tool usage to promote database changes through the database environments
3. Provide all database passwords by instance to include SYS, SYSTEM, SYS DBA and the UNIX Oracle passwords.
4. Provide documentation on current database management processes utilized to manage the development, test and production instances.
5. Document or briefly outline the current database development model, including environments for development/testing/production, how database changes and code are migrated through the development model.
6. Document instance backup procedures currently in place with assistance of the system administrator.
7. Document UNIX batch scripts for all instances with the assistance of the system administrator.
8. Document strategy for final implementation of indexing, unique keys, referential integrity keys, etc. to the database at the latest release.
9. Document strategy and prototypes of database security implementations, including but not limited to single-sign-on and LDAP usage.
10. Document data loading/conversion scripts for all instances with assistance of data conversion individual.

6. ORACLE DATABASE SETUP PARAMETERS

Having understood what is involved in the Environment Setup for such a project let us follow-up by understanding the nuts-and-bolts of the actual Oracle Instance installed in your project. Let us begin by checking out the initialization parameters – i.e. the ***init.ora*** file contents.

There should exist a quick and easy way of comparing parameters between two instances while ignoring the obviously different parameters.

```
--@paramcompare.sql
spool paramcompare_ProjDEV3_ProjTST1.lst

set lines 1000
set trimspool on

-- v$system_parameter
WITH zzz as
(select a.name as PARNAME
 from v$system_parameter@ProjDEV3 a
    , v$system_parameter@ProjTST1 b
 where trim(a.name) = trim(b.name)
 and a.value <> b.value
)
select substr(z.parname,1,32) PARAM_NAME
, substr(a.value,1,50) ProjDEV3
, substr(b.value,1,50) ProjTST1
from zzz z
, v$system_parameter@ProjDEV3 a
, v$system_parameter@ProjTST1 b
where a.name = z.parname
and b.name = z.parname
and z.parname not in (  'audit_file_dest'
                      , 'background_dump_dest'
                      , 'control_files'
                      , 'core_dump_dest'
                      , 'db_name'
                      , 'dispatchers'
                      , 'instance_name'
                      , 'log_archive_dest_1'
                      , 'rdbms_server_dn'
                      , 'service_names'
                      , 'user_dump_dest'        )
order by 1;


-- v$system_parameter2
WITH zzz as
(select a.name as PAR2NAME
```

```
 from v$system_parameter2@ProjDEV3 a
    , v$system_parameter2@ProjTST1 b
 where trim(a.name) = trim(b.name)
 and a.value <> b.value
)
select substr(z.par2name,1,32) PARAM2_NAME
, substr(a.value,1,50) ProjDEV3
, substr(b.value,1,50) ProjTST1
from zzz z
, v$system_parameter2@ProjDEV3 a
, v$system_parameter2@ProjTST1 b
where a.name = z.par2name
and b.name = z.par2name
and z.par2name not in ( 'audit_file_dest'
                      , 'background_dump_dest'
                      , 'control_files'
                      , 'core_dump_dest'
                      , 'db_name'
                      , 'dispatchers'
                      , 'instance_name'
                      , 'log_archive_dest_1'
                      , 'rdbms_server_dn'
                      , 'service_names'
                      , 'user_dump_dest'        )
order by 1;

-- v$spparameter
WITH zzz as
(select a.name as SPPARNAME
 from v$spparameter@ProjDEV3 a
    , v$spparameter@ProjTST1 b
where trim(a.name) = trim(b.name)
and a.value <> b.value
)
select substr(z.spparname,1,32) SPPARAM_NAME
, substr(a.value,1,50) ProjDEV3
, substr(b.value,1,50) ProjTST1
from zzz z
, v$spparameter@ProjDEV3 a
, v$spparameter@ProjTST1 b
where a.name = z.spparname
and b.name = z.spparname
and z.spparname not in (  'audit_file_dest'
                       , 'background_dump_dest'
```

```
                          , 'control_files'
                          , 'core_dump_dest'
                          , 'db_name'
                          , 'dispatchers'
                          , 'instance_name'
                          , 'log_archive_dest_1'
                          , 'rdbms_server_dn'
                          , 'service_names'
                          , 'user_dump_dest'        )
    order by 1;


    spool off;
```

```
PARAM_NAME                      ProjDEV3         ProjPRD2
------------------------------- -------------- ------------
aq_tm_processes                 2              0
compatible                      10.1.0.2.0     9.2.0
cpu_count                       2              6
db_file_multiblock_read_count   37             8
db_unique_name                  bmvdev3        bmvprd2
job_queue_processes             12             1
log_archive_format              %t_%s_%r.dbf   %t_%s.dbf
log_buffer                      524288         786432
log_checkpoint_interval         1              0
log_checkpoint_timeout          0              1800
parallel_max_servers            40             120
shared_servers                  1              10
sp_name                         bmvdev3        bmvprd2

16 rows selected.


PARAM2_NAME                     ProjDEV3         ProjPRD2
```

```
-------------------------------- -------------- ------------
aq_tm_processes                  2              0
compatible                       10.1.0.2.0     9.2.0
cpu_count                        2              6
db_file_multiblock_read_count    37             8
db_unique_name                   bmvdev3        bmvprd2
job_queue_processes              12             1
log_archive_format               %t_%s_%r.dbf   %t_%s.dbf
log_buffer                       524288         786432
log_checkpoint_interval          1              0
log_checkpoint_timeout           0              1800
parallel_max_servers             40             120
shared_servers                   1              10
sp_name                          bmvdev3        bmvprd2

16 rows selected.


no rows selected
```

7. DATABASE VERSIONING

Ok – I have understood a bit about the init.ora. Can I know how we actually version the database and/or its individual objects?

There is no standard way observed for versioning databases. Here is a quick run-down on the two most popular ***home-grown*** methods of versioning the database and implementing the logic in the checked-in checked-out code scripts using general purpose version control toolkits like Microsoft VSS (Visual SourceSafe) or Rational ClearCase or CVS (Concurrent Version Systems).

7.1. INCREMENTAL VERSIONING

In this technique

- For Pl/Sql based objects like Packages/Procedures/Triggers/etc check-in and check-out the entire object as one script.
- For Tables/Views/Indexes/etc, initially check-in the entire CREATE script, thereafter check-in only the ALTER object scripts.

7.2. COMPLETE VERSIONING

In this technique

- For PL/Sql based objects like Packages/Procedures/Triggers/etc check-in and check-out the entire object as one script.
- For Tables/Views/Indexes/etc, initially check-in the entire CREATE script, thereafter also, check-in the entire CREATE script as one. Actual implementation will have a DROP table script preceding a CREATE table script.

7.3. PROS/CONS

- In the incremental versioning technique, propagation of code changes is easier from DEV to PROD environments. However, if you need to build an entirely new instance/db, you have to implement many more scripts.
- In the latter method, new instance/db can be created easily, but we loose the ability of selectively retaining specific versions (ALTERS).
- One could implement a hybrid approach in which we do incremental releases till a point-in-time and at an appropriate major release-point do a complete base-lining one again.

- In both instances you will have some table structure maintaining the release-version information in the database.

Please note that in addition there are a number of Oracle (Change Management Pack) and 3rd party tools (Erwin, DBArtisan, etc) that allow doing one or more of the following:

- Reverse Engineer database and Schema Definitions
- Capture and version baselines
- Compare Databases and schemas, or baselines
- Copy database objects with data, with a subset of the data, or without data
- Update database object definitions (ALTER TABLE)

8. CODE REVIEWS

While the DBA needs to know the setup and init.ora – why is he so interested in the sql-code that we (developers) check-in? And what is actually done during Database Code-Reviews? Is it not sufficient that the query returns what it is supposed to return?

A typical DBA spends a majority of his time during Project Development Phase to spread awareness about good coding practices. Here are some items that must be understood, preached and followed in order that good code is deployed to Production.

- Rows Returned
  - Minimize the number of rows searched and/or returned
  - Code all necessary predicates to limit the result to only the rows needed
  - Avoid generic queries that do not have a WHERE clause
- Column Selection
  - Minimize the number of columns retrieved and/or updated
  - Specify in SELECT only the columns needed for the business logic at hand
  - Avoid *SELECT * from*
  - Extra columns increases row size of the result set
  - Retrieving very few columns can encourage index-only access
- Avoid Sorting

  - DISTINCT - always results in a sort
  - UNION - always results in a sort
  - UNION ALL - does not sort, but retains any duplicates
  - ORDER BY
    - may be faster if columns are indexed
    - use it to guarantee the sequence of the data
  - GROUP BY
    - specify only columns that need to be grouped
    - may be faster if the columns are indexed
  - do not include extra columns in SELECT list or GROUP BY because Oracle must sort the rows

- SubQueries
  - Oracle processes the subquery (inner query) first before the outer query
  - You may be able to improve performance of complex queries by coding a complex predicate in a subquery
  - Applying the predicate in the subquery may reduce the number of rows returned
- Use Inline Views
  - Inline views allow the FROM clause of a SELECT statement to contain another SELECT statement
  - May enhance performance of the outer select by applying predicates in the inner select
  - Useful when detail and aggregated data must be returned in a single query
- Join Predicates
  - Response time -> determined mostly by the number of rows participating in the join
  - Provide accurate join predicates
  - Never use a JOIN without a predicate
  - Join ON indexed columns

9. EXPLAIN PLANS – DOCUMENT THEM

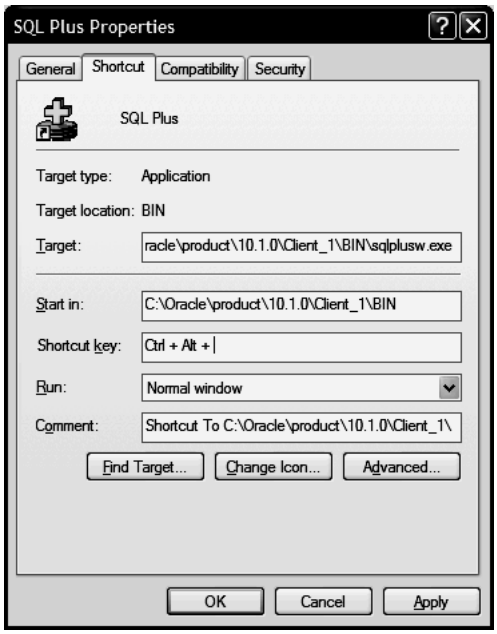
Oh! Explain-plans!! What has the developer got to do with it? That is the DBA's job! Why document them?

A typical DBA could spend a majority of his time during Project Test/Acceptance Phase in tuning inefficient sqls and this may lead to project slippage. Encourage the developers to generate the Explain Plans during the Development and Unit Testing phase itself so as to save time later. Document them to establish performance yardsticks that would be valuable to evaluate performances in higher environments as well as creating a decision support vehicle to making system-side modifications in say certain init-ora parameters (which we have already touched up upon in bullet #6 above)

- Directory is …\Source_Code_VOB\project\ExplainPlan
- Two files needed
- Preferred method of generating explain plans --- SQL*Plus
- <*script*>.*sql* ---- this will contain the sql or the couple of sqls you used for comparing
- <*script*>.*lst* ----- this will contain the explain plans
- You may need to replace any input parameters '?' with some real-looking values.
- ExplainPlanTemplate.sql

```
set echo on
set trimspool on
set lines 999
set pages 0

spool <put the script name here>.lst

-- Original Query
explain plan for
<put the actual sql here>
;

select * from table(dbms_xplan.display);

spool off;
```

- Where did my spooled .lst file go?
  - When you use the SQL*Plus QuickLink from the Windows-Start Menu, you are actually executing the following executable C:\oracle\..\..\..\bin\sqlplusw.exe
  - The default StartIn directory for this shortcut points to directory C:\oracle\..\..\..\bin
  - Hence your spooled file automatically goes to the StartIn directory defined above.



  - To send the spool files to a directory of your choice say C:\sql do the following
    - Create a shortcut to the sqlplusw.exe
    - Copy the shortcut to the desktop
    - Change the StartIn directory to C:\sql and click Apply

10. EXPLAIN-PLAN OPERATIONS AND COST EXPLAINED

10.1. B*TREE INDEX ACCESS PATHS

- INDEX UNIQUE SCAN
  - Equality predicate on unique or primary key column(s)
  - Generally considered most efficient access path

- usually no more than 3-4 buffer gets
- If table is "small", FULL TABLE SCAN could be cheaper

- INDEX RANGE SCAN
  - Equality predicate on non-unique index, incompletely specified unique index, or range predicate on unique index
  - Be careful of the size of the range
  - Large ranges could amount to huge # of buffer gets
  - If so, consider a FAST FULL INDEX SCAN or FULL TABLE SCAN

- **INDEX FULL SCAN**
  - Will scan entire index by walking tree, in index order
  - Provides ordered output, can be used to avoid sorts for ORDER BY clauses that specify index column order
  - Slower than INDEX FAST FULL SCAN, if there is no ORDER BY requirement

- INDEX FAST FULL SCAN
  - Will read index, in disk block order, and discard root and branch blocks.
  - Equivalent to FULL TABLE SCAN, for an index
  - Fastest way to read entire contents of an index

- INDEX SKIP SCAN
  - Allows some benefits of multi-column index, even without specifying the leading edge.
  - Oracle will "skip scan", starting with root block, skipping through B*-tree structure, masking sections of tree that cannot have applicable data

10.2. BITMAP INDEX ACCESS PATHS

- BITMAP INDEX SINGLE VALUE
- BITMAP INDEX RANGE SCAN
- **BITMAP INDEX FULL SCAN**
- BITMAP AND

- BITMAP OR
- BITMAP NOT
- BITMAP CONVERSION COUNT
- BITMAP CONVERSION TO ROWIDs

10.3. OTHER ACCESS PATHS

- UNION, UNION-ALL, MINUS, INTERSECTION
  - Directly correspond to the SQL set operators
  - UNION-ALL is cheapest, since no SORT(UNIQUE) is required
- **TABLE FULL SCAN**
  - Reads all blocks allocated to table, up to the High Water Mark
  - Can be most efficient access path for "small" tables
  - Can cause significant physical I/O, particularly on larger tables
- TABLE ACCESS BY ROWID
  - Generally used in conjunction with an index access path, where rowid has been identified, but Oracle needs access to a column not in the index
  - Consider whether adding a column to an existing index will provide substantial benefit (Don't forget to consider the additional overhead that will be incurred on every DML involving the index)
  - Cost is directly proportional to number of rowid lookups that are required
- TABLE (HASH)
  - More efficient than index access

10.4. JOIN METHODS

- Nested Loops
  - Generally geared towards FIRST_ROWS access
  - Ideal for B*Tree index driven access, small row sources
  - When this is the case, can't be beat for first row response time
  - Can get very costly very quickly if no index path exists or index path is inefficient
- **Sort Merge**
  - Generally Geared towards ALL_ROWS access

- Can be useful for joining small to medium size row sources, particularly if viable index path is not available or if Cartesian join is desired

- Hash
  - Most controversial (and misunderstood) join method
  - Can be very powerful, when applied correctly
  - Useful for joining small to medium sized row source to a large row source

10.5. WHAT IS THE COST IN AN EXPLAIN PLAN?

Asktom.oracle.com

- ✓ *cost is not misleading*
- ✓ *cost is just a number, a number that is the result of an algorithm*
- ✓ *an algorithm that takes many inputs, many parameters*
- ✓ *an algorithm that given all of the inputs and parameters returns the query plan with the lowest cost for that particular query*
- ✓ *an algorithm that if the inputs are correct, the parameters good -- returns the query plan that should run the fastest.*
- ✓ *an algorithm implemented by mere humans, so.....*

11. ORACLE META-DATA TABLES

What are all the secret tables the DBA's always are querying against? They have weird looking names that begin with ALL_ or DBA_.

OR

Why does a search on PERSON_NAME not do a Full-Table-Scan, but similar query on ORGANIZATION_NAME do it? Are the indexes set differently on those tables? How can I find out the answer to this myself?

- The ALL_ tables
  - ALL_TABLES: Stores information on all tables you have access to.
  - ALL_TAB_COLUMNS: Stores information on columns of all tables you have access to.
  - ALL_INDEXES
  - ALL_IND_COLUMNS

```
suhas@projdev2> select ai.table_name
  2    , ai.index_type
  3    , ai.index_name
  4    , substr(aic.column_name,1,30) column_name
  5    from all_indexes ai
  6    join all_ind_columns aic on (ai.index_name = aic.index_name)
  7    where ai.owner = 'PROJ_OWNER'
  8    and ai.table_name in ('PERSON_NAME', 'ORGANIZATION_NAME')
  9    order by 1,2 desc
 10  ;


TABLE_NAME           INDEX_TYPE                INDEX_NAME           COLUMN_NAME
-------------------- ------------------------- -------------------- --------------------
ORGANIZATION_NAME    NORMAL                    ORGANIZATION_NAME_IF CUSTOMER_ID
ORGANIZATION_NAME    NORMAL                    ORGANIZATION_NAME_PK ORGANIZATION_NAME_ID
PERSON_NAME          NORMAL                    PERSON_NAME_PK       PERSON_NAME_ID
PERSON_NAME          NORMAL                    PERSON_NAME_IDX1     CUSTOMER_ID
PERSON_NAME          FUNCTION-BASED NORMAL     PERSON_NAME_FX1      UPPER("LAST_NAME")
PERSON_NAME          FUNCTION-BASED NORMAL     PERSON_NAME_FX1      UPPER("FIRST_NAME")
PERSON_NAME          FUNCTION-BASED NORMAL     PERSON_NAME_FX1      UPPER("MIDDLE_NAME")

7 rows selected.
```

- Thus we know that PERSON_NAME table has some functional indexes in addition to its regular Primary and Foreign-Key indexes – it's those functional indexes which have avoided the quite expensive FTS.

12. SAMPLE DATABASE ARCHITECTURE – A BIRDS EYE VIEW

- *What are the specifications of the UNIX machines that house our databases?*

Primarily we use four Unix machines to house our databases. They are as follows:

Sr. No.	Box Name	Configuration	Instances held currently	Oracle Version
1.	DBDEV	2 processors 4 Gig	ProjDEV1 ProjDEV2	Oracle9i Enterprise Edition Release 9.2.0.4.0
2.	DBTEST	2 processors 4 Gig	ProjTST1 ProjTST2	-- as above --
3.	PRDDB1	6 processors 8 Gig	ProjPERF ProjPRD1	-- as above --

4.	PRDDB2	6 processors 8 Gig	ProjPRD2	Oracle Database 10g Enterprise Edition Release 10.1.0.2.0

- *What are the major schemas of the various Project databases from a Data Architecture perspective?*

INSTANCE	SCHEMA	What lives here?	RELEASE	ACTIVITY
ProjDEV1	Proj_OWNER	Proj Application Owner Objects	8.19	Development Instance1
9.2.0.4	Proj_COMMON	Proj Common Objects		
	Proj_SECURITY	Proj Security Objects		
	Proj_QUEUE	Proj Queue Objects		
ProjDEV2	Proj_OWNER	Proj Application Owner	8.17	Development Instance2
9.2.0.4	Proj_COMMON	Proj Common Objects		
	Proj_SECURITY	Proj Security Objects		
	Proj_QUEUE	Proj Queue Objects		
ProjTST1	Proj_OWNER	Proj Application Owner	8.17	System Test Instance 1
9.2.0.4	Proj_COMMON	Proj Common Objects		
	Proj_SECURITY	Proj Security Objects		
	Proj_QUEUE	Proj Queue Objects		
ProjTST2	Proj_OWNER	Proj Application Owner	8.17	System Test Instance 2
9.2.0.4	Proj_COMMON	Proj Common Objects		
	Proj_SECURITY	Proj Security Objects		
	Proj_QUEUE	Proj Queue Objects		
ProjPERF	CONVERT1	(has sample conversion data)		Data Conversion
9.2.0.4	CONVERT2	(has complete conversion data)		Performance Testing
ProjPRD1	Proj_OWNER	Proj Application Owner Objects	8.17	Production Instance1
9.2.0.4	Proj_COMMON	Proj Common Objects		
	Proj_SECURITY	Proj Security Objects		
	Proj_QUEUE	Proj Queue Objects		
ProjPRD2	Proj_OWNER	Proj Application Owner Objects	8.15	Production Instance 2
10.1.0.2	Proj_COMMON	Proj Common Objects		Proj Security Policy Testing
	Proj_SECURITY	Proj Security Objects		LDAP Testing
	Proj_QUEUE	Proj Queue Objects		**Future Production database**

- *What are the typical objects in these schemas (say ProjDEV2)?*

OWNER	TABLE	INDEX	FUNCTION	PACKAGE	PROCEDURE	SEQUENCE	SYNONYM	TRIGGER	VIEW
Proj_COMMON	6	2	6	1	14	0	1416	4	1
Proj_OWNER	**751**	**3874**	**8**	**11**	**1**	**515**	**3**	**367**	**14**
Proj_QUEUE	3	9	0	0	0	0	0	6	3
Proj_SECURITY	**3**	9	0	0	0	1	1330	3	0
Proj_WEB_USER	0	0	0	0	0	0	**1518**	0	0

- *How are these objects documented?*

  The Proj uses the tool ErWin to document the database model.

- *What are sequences used for in the Project?*

  Primary keys could either be 'intelligent' i.e. their actual values imply some data logic (e.g. NY for New York), or 'surrogate' i.e. their actual values are merely numbers devoid of any data logic. All primary keys in the Project database are surrogate in nature. Suitable 'sequence' objects are used to provide the primary key values. All such sequence objects are named identical to the column they provide data for – e.g. the customer_id field takes the value from the customer_seq sequence.

- *What are the naming conventions used in Project database objects?*

Lookup Tables	LU_
Lookup Filter Tables	LUF_
Staging Tables	STAGE_
Security Tables	SEC_
All Surrogate PK's	_ID
Sequence Names	_SEQ
Views	_V
Constraints --Primary Key	_PK
Constraints – Foreign Key	_FK
Constraints – Unique Key	_UK
Constraints – Check	_CK
Indexes – Regular	_I

Indexes – For Foreign Keys	_IF
Indexes – Bitmapped	_IB
Indexes – Functional	_FX
Triggers	_TR

- *How are DB versions maintained?*

  Unlike Application code, all database object structures/data cannot be checked in and out in ClearCase and versioned/labeled as such. Maintaining versions of specific database objects like Packages, Procedures, Functions and Triggers can be done this way and is maintained as such. However overall where do you go to find the exact version a specific database is at? There is a table named VERSION (in the Proj_OWNER schema) which contains all information about the release the current instance is at.

```
suhas@projdev2> select * from version;

VERSION_NUM ITERATION  VERSION_DESCRIPTION        SUBVERSION_NUM MOD_DATE
----------- ---------  ------------------------- -------------- ---------
8.17         8         Iteration 8 Eighth Release              1 02-AUG-04
```

13. SAMPLE DATABASE ARCHIVAL STRATEGY

How are all our databases backed up? Where are the backups kept and how long are they retained?

The databases are FULLy (not incremental!) backed up routinely and archived on tape. The tapes are on a daily system with the Disaster Recovery plan being storage of tapes with 'Iron Mountain' – a third party contract that is responsible for offsite storage. Our contact is a Ms. xxx at yyy. While the actual database backups are done via script located at "/oracle/dba_stuff/restore.sh"

- A quick look at crontab will give exact timings, but in general we do
  - An PROD export at 18:00 hrs (this is end-of-day snapshot before running nightly reports)
  - Another PROD export at 05:00 hrs (lesser known to general audience)
  - All databases exported sometime between 20:00 hrs and 23:00 hrs daily.
  - ANALYZE on all databases at 20:00 hrs
  - Purge archive (> 3 days old) at 06:00 hrs

14. SAMPLE HANDOVER CHECKLIST

I think I am aware of much more now about the hand-over. Is there a complete hand-over documentation checklist that we should go over?

Database Instances/Release versions and IP map	☐
Unix Database setup	☐
Tasks involved in refreshing from PROD	☐
Database Archival Strategy	☐
Data Migration Support	☐
Oracle Report Server configuration	☐
Database Version Control process	☐
WebLogic/Cocobase related tables	☐
MasterCode Tables Source/Maintenance	☐
HISTORY Schema Explanation	☐
<Specific> Schema and Objects	☐
Roles and Profiles	☐
Misc. files and their locations	☐
Security Policies implemented	☐
Important Contact Names and Numbers	☐

15. ADDITIONAL ASPECTS THAT CAN BE ADDED TO THE HAND-OVER DOCUMENT

I am sure that will not cover everything that is there in a project. What about other projects which could have varying demands on technology and hence disparate solutions?

Absolutely! The above is just a starting point – please feel free to add more sections on some/all of the below given points:

- Database Crashes in the recent past – causes and their rectification
- Granting developers the ability to Kill their own sessions
- Setting up public/private synonyms
- Creating/Cloning databases
- DBLinks – Oracle/Heterogeneous connectivity
- Materialized Views / Partitioned tables
- RAC
- RMAN
- Oracle Patch Releases
- Regulatory requirements – SOX compliance
- Environment Specific pointers (TEST/QA)
- End Users / Stakeholders / Clients

16. SUMMARY

We started out with a quest -- *What must the Database Administration/Architecture hand-over document say and what should the actual handover entail?* We have walked through such a requirement and equipped ourselves with the necessary and sufficient tools to takeover a DBA's role. The session takes baby steps and covers a large multitude of topics that are relevant for hand-over of one of the most crucial of Oracle Job Descriptions to date. However, given the vastness of the various technologies within Oracle and the many years of real-life experience it takes for someone to grasp them and become a good Oracle-DBA – this session is in no way a promise to be a silver-bullet to give you all that and make you ready for the tough challenges in such a short time. Please feel free to use the check-list as your own starting point and keep enhancing it so when the true disaster strikes – your team will be ready. A well documented transition plan will not only guarantee project success, but also a smooth transition-in for the new person.

17. ACKNOWLEDGMENTS AND RESOURCES

1. www.asktom.com