HOW TO IMPROVE YOUR ORACLE CAREER, AND MAKE MAGIC WITH ORACLE

Introduction

The main question about the DBA job I hear all the time is: How can I became a successful DBA?

Most of the people that I talk to who have difficulties starting out in their DBA career really have an issue trying to absorb the mountainous volumes of information that a DBA needs to know this days. The evolution of the DBA role in the past few years was amazing, from a role that was basically responsible for administrate one or two small Oracle Databases and interact very closely with some System Administrators to become a super role (most of the time absorbing System and Network administrator responsibilities) some modern DBA responsibilities could be to manage:

- Several Oracle Databases and Data Warehouses
- High Availability environments like RAC and Standby Databases
- Other type of RDBMS (MySQL, SQL Server, DB2, etc)
- Support servers (Application and DB)
- Security and Network stability
- Storages and Clusters
- Mentor other DBAs
- Backup and Recovery Strategy
- Handle User problems (including functional side of applications)
- Review SQL and PL/SQL codes
- Control and execute promotions to production environments

As per example, here are some common duties for a DBA in today's world:

- Monitor database instances on a daily basis to ensure availability.
- Resolve unavailability issues.
- Collect system statistics and performance data for trending and configuration analysis.
- Configure and tune DB instances for optimal performance under application specific guidelines.
- Analyze and administer DB security. Control and monitor user access. Audit DB usage when necessary.
- Monitor backup procedures and Provide recovery when needed.
- Develop and test backup and recovery procedures.
- Upgrade RDBMS software and apply patches when needed.
- Upgrade or migrate database instances as necessary.
- Support application developers with any and all dB related activities.
- Keep up with DB trends & technologies.
- Use new technologies when applicable.

- Install, test, and evaluate new Oracle related products.
- Perform storage and physical design.
- Balance design issues to achieve optimal performance.
- Create, configure and design new DB instances.
- Diagnose, troubleshoot and resolve any DB related problems.
- Work with Oracle Support if necessary to bring problems to a successful resolution.
- Ensure that Oracle networking software is configured and running properly.
- Work with System Administrators (UNIX & NT) to ensure Oracle related matters are handled properly, or in some cases, do yourself it.
- Train new DBA's
- Create, Manage and Monitor Standby Databases
- Understand your user Applications and needs.
- Configure and Manage Database and Application Servers
- Manage and configure Clusters , DWs and AS
- Configure and Manage different type of RDBMS
- XML, Java, PHP, HTML, Linux, Unix, Windows Scripting
- Create any necessary scripts for effective and occasionally periodic dB maintenance activities.
- Capacity Planning /Hardware Planning

Like you can easily see, the DBA job is not easy, and each professional need to be capable to be multitasks and manages a lot of responsibilities and stress.

In this paper, we will see several examples on how to improve your DBA career and how to become a real success DBA.

First, Learn to change yourself

If you want to become a successful professional, first you need to educate yourself to be successful! Your future success depends only in your attitude today. You control your life, nobody else!

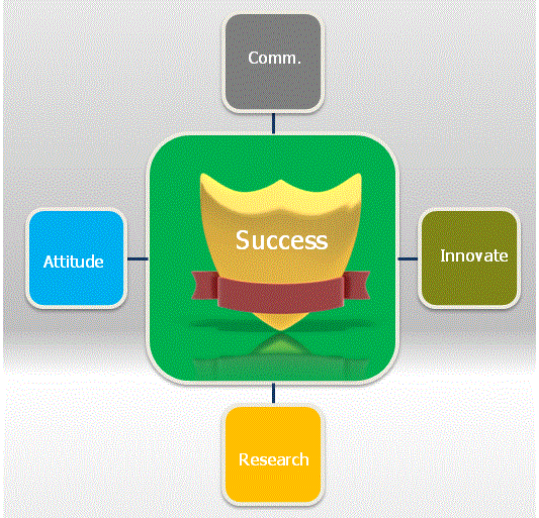

Figure 1

Becoming a successful DBA is a combination of:

- Your professional attitude, always think positive and always look for solutions instead to kill yourself in a cup of water.
- Learn how to research, before do something, investigate, search in the internet, read manuals. You need to show that you know how to do a properly research and look for solutions for your problems yourself.
- Innovate, don't wait for others to do your job, or because the other DBAs don't care about the business you will do the same. Learn to innovate, learn to become a leader and make everyone follow your example with results. Think Different!
- Learn to communicate properly; the best way to learn how to communicate effectively is learning to listen first. Listen, than analyze the context expressed and only than communicate an answer in a professional and honest way to your peers. Always treat everyone the same way you would like to be treated.

Albert Einstein said one time:

"If I had one hour to save the world, I would spend fifty-five minutes defining the problem and only five minutes finding the solution"

Learning to be Proactive

Why check the problems only when they are critical, or when is too late and the database is down, or the users are screaming?

Being proactive is the best approach to keep your DB healthy and to show your company, or your clients that you really care about them.

Many DBA's expend most of their time being firefighters only, fixing problems and working on user's requests all the time. They don't do any proactive work; this mentality only will cause an overload of work to them, thousands of dollars of overtime, several hours without access to the data to the users, poor performance to the applications, and what is worse of all, several unhappy users thinking that you doesn't have the knowledge needed to take care of their data.

Let's mention a small example, you have the archive log area alert set to fire when it is 95% full, and this happens in the middle of the night, some DBA's will take seriously the alert and solve the problem quickly, others will wait until the next day to take care of it because they are tired, or sleeping, or they are in a place without internet access at the moment the alert arrived. Will be a lot easier if they set a proactive alert to be fire when 75% or 85%, or even better, take a look in the general health status of the DB before leave their work shift, to try to detect and solve any possible problem before be a real problem and be awake in the middle of the night or during the weekend (Remember how important is your personal and family time). I'll always recommend to DBA's to run 2 checklists daily, one in the start of their shift and other before they leave their shift.

I know several DBA's that complain all the time that they got so many calls when they are on call, but they don't do anything to solve the root problem, they only expend their time to solve the symptoms.

In my blog you can find an Oracle checklist script that will help to make your life a little easier (This is not my complete script, but will be a good start for you). This script is a compilation of several normal checklists and you can setup them with your own requirement and thresholds and always remember to have a baseline to compare. This script will not only help you to detect future or current problems, but also will help you to detect possible tuning requirement.

Here is an example of the script first phase outcome:

```
- ————————————————————— -
- Oracle Instance Information
- ————————————————————— -
Cpu_Count                        4   |      Host_Name             OLIVER
Instance_Name                 prod   |      Database_Status       ACTIVE
Status                        OPEN   |      Startup_Time          10-01-2009 19:50
Version                 11.1.0.7.0   |      Instance_Role         PRIMARY_INSTANCE
Database Space (Mb)           36604    |      SGA (Mb)              511
Nb. Datafiles                   43    |      Nb. Tempfiles         1

Archive destination LOCATION=E:\oracle\oradata\prod\archive
Database log mode ARCHIVELOG
Background Dump Dest d:\oracle\diag\rdbms\prod\prod\trace
Spfile D:\ORACLE\PRODUCT\11.1\PROD\DATABASE\SPFILEPROD.ORA

Redo size (Kb) 102400
- ————————————————————— -
- Instance CheckList -
- ————————————————————— -
Instance Status                  OK    |      Listener Status      OK
- ————————————————————— -
- Performance Memory CheckList -
- ————————————————————— -

Total Sessions < 700             OK - 19
Active sessions number <15       OK - 9
Data Buffer Hit Ratio > 80       OK - 97
L.Buffer Reload Pin Ratio > 99   OK - 99
Row Cache Miss Ratio < 0.015     NO - 1.351
Dict.Buffer Hit Ratio > 80       OK - 99
Log Buffer Waits = 0             NO - 110
Log Buffer Retries < 0.0010      OK - 0
Switch number (Daily Avg) < 5    OK - 1
Jobs Broken = 0                  OK - 0
Shared_Pool Failure = 0          OK - 0

- ————————————————————— -
- Storage CheckList                                  -
- ————————————————————— -

Dba_Tablespaces Status           OK   |      V$Log Status          OK
V$Datafile Status                OK   |      V$Tempfile Status    OK
V$Recover_File                   OK   |      V$Recovery_Log       OK
Tablespace in Backup Mode = 0    OK - 0
Tablespace < 95%                 OK - 0
Objects Invalid = 0              NO - 147
Indexes unusable = 0             OK - 0
Trigger Disabled = 0             NO - 5
Constraint Disabled = 0          NO - 2
Objects close max extents = 0    OK - 0
Objects can not extent = 0       NO - 552
User Objects on Systems = 0      NO - 26
FK Without Index = 0             NO - 138

- ————————————————————— -
- Datagard CheckList                                -
- ————————————————————— -

Datagard Errors = 0              OK - 0
Datagard Gap = 0                 OK - 0
Archives not Aplied < 5          OK - 2
```

```
- ————————————————————- -
- Installed options :
- ————————————————————- -
- Objects option
- Connection multiplexing option
- Connection pooling option
- Database queuing option
- Incremental backup and recovery option
- Instead-of triggers option
- Parallel load option
- Proxy authentication/authorization option
- Plan Stability option
- Coalesce Index option
- Transparent Application Failover option
- Sample Scan option
- Java option
- OLAP Window Functions option
```

You also have several tools available in the market that can help you to monitor and setup your DB alerts, and help you with the proactive monitoring like: Grid Control, Enterprise Manager, Insider (FourthElephant), Spotlight (Quest) or if you prefer, your own scripts. The idea is to use them always on a proactive way, never reactive.

Let's change our mentality, let stop being a firefighter and start to be a real hero!

Backup And Recovery

It's time to be "Proactive with Backup & Recovery", always when I arrive on a new client I ask the DBA on charge the following questions:

- Do you have your recovery strategy documented step by step?
- Are you 100% sure that your tape backups are usable?
- Do you know exactly how long a recovery on your production environment will take if necessary?

And almost 90% of the time the answers will be:

- No!
- I not sure, but I think so!
- No idea, probably…!

You will be on shock to know how many times I'm call to support a DBA to try to recover a Database because the most current tape backup is unusable!

Backup & Recovery are a very important (crucial) part of a DBA role, as a DBA I'll never be stressed enough to repeat over and over what in my opinion is the most important rule for a DBA:

"The most important rule with respect to data is to never put yourself into an unrecoverable situation, never!"

You know, because bad stuff happens….



…When you less expect, and due to this, I'll always recommend a DBA to perform a proactive approach to his/her Database Backup and Recovery strategy.

The main idea is:

- Randomly choose a backup tape and recovery it on a test machine (It can be a virtual one).
- Take this opportunity to document all the recover process.
- Review the entire process ant try to improve it!
- Repeat this exercise every month and try to involve other DBAs in the process!

This easy process will allow you to:

- Test your Tape backups and see if they are being backup correctly.

- Check and improve your recovery knowledge and strategy.
- Document all your recovery process that could be used for any other DBA in the company in case you are not available in the recovery situation.
- Detect any error on your backup & recovery strategy.
- Know your recovery time. Next time your manager asks you" Do you know how long a recovery will take? You will know the exact answer.
- Have an opportunity to review your process and try to make it more efficient.

Like you can see, this is an easy proactive exercise that will allow you and your company to be prepared in case of a disaster and recovery situations occurs, and you know when this always happens….

Making Magic with Datapump

A lot of people don't know several powerful functionalities that we have available when using Data Pump (expdp/impdp), most of the people only use these tools to export and import data (in other words, only to move data), and never notice that it can be used for example to help us to do:

- Data Masking
- Build a Metadata Repository
- Create a version control
- Clone Users (Create a new user using and existent user as a template)
- Create smaller copies of production
- Create your database in a different file structure
- Move all objects from one tablespace to another
- Move a object to a different schema (A simple example, change a table owner)

Now let's see how each functionality I mentioned above can be used at real life.

1) Data Masking

In many organizations (I hope so) the DBA's had the obligation for a security and compliance purpose to mask all sensible information that leaves the production environment to as an example to refresh or create a QA/Test or Dev environment. To help us to address this kind of requirements we could easily use the Enterprise Manager Data Masking Pack (remember it is an extra pack, and consequently you need to pay extra to use it), or as a different option, use the "remap_data" parameter available in Data Pump to help you with this requirement(**this is a new functionality at 11g)!

Let's use the classic SSN (Social Security Number) example to illustrate how it works:

a) First let's create the table for the test and load some data on it.

```
SQL> CREATE TABLE HR.EMPLOYEE
  2  ( EMP_ID   NUMBER(10) NOT NULL,
  3    EMP_NAME VARCHAR2(30),
  4    EMP_SSN  VARCHAR2(9),
  5    EMP_DOB  DATE
  6* )
SQL> /

insert into hr.employee values (101,'Francisco Munoz',123456789,'30-DEC-73');
insert into hr.employee values (102,'Horacio Miranda',234567890,'17-JUL-76');
insert into hr.employee values (103,'Evelyn Aghemio',659812831,'02-OCT-79');
```

b) The second step will be to create the remap function:

```
SQL> create or replace package pkg_masking
   2 as
   3 function mask_ssn (p_in varchar2) return varchar2;
   4 end;
   5 /
SQL> create or replace package body pkg_masking
   2 as
   3 function mask_ssn (p_in varchar2)
   4 return varchar2
   5 is
   6 begin
   7 return lpad (
   8 round(dbms_random.value (001000000,999999999)),9,0);
   9 end;
  10 end;
  11 /
```

This function will take a varchar argument and returns a 9 char. We will use this function to mask all SSN information inside our employee table.

```
SQL> desc employee
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
EMP_ID                                    NOT NULL NUMBER(10)
EMP_NAME                                           VARCHAR2(30)
EMP_SSN                                            VARCHAR2(9)
EMP_DOB                                            DATE

SQL> select * from employee;

EMP_ID     EMP_NAME                       EMP_SSN   EMP_DOB
---------- ------------------------------ --------- ---------
101        Francisco Munoz                123456789 30-DEC-73
102        Horacio Miranda                234567890 17-JUL-76
103        Evelyn Aghemio                 345678901 02-OCT-79
```

For this example, all you want to mask is the column EMP_SSN, which contains the SSN of each employee.

b) Now we are going to export the table employees using the expdp tool, and while exporting, we will use the parameter "remap_data" to mask the data for us in the dump file using the function we previously created.

```
$expdp hr/hr tables=hr.employee dumpfile=mask_ssn.dmp directory=datapump
remap_data=hr.employee.emp_ssn:pkg_masking.mask_ssn
```

Note: By defect the "remap_data" parameter will use the user doing the export as the owner of the remap function, if the schema owner of the function is different you will need to use the following commad:

```
$ expdp hr/hr tables=hr.employee dumpfile=mask_ssn.dmp directory=datapump
remap_data=hr.employee.emp_ssn:owner.pkg_masking.mask_ssn
```

c) Now all we need to do is to import the mask_ssn.dmp in our QA/Test or Dev Database and it will magically have the new values there.

```
SQL> select * from employee;

EMP_ID     EMP_NAME                       EMP_SSN   EMP_DOB
---------- ------------------------------ --------- ---------
101        Francisco Munoz                108035616 30-DEC-73
102        Horacio Miranda                324184688 17-JUL-76
103        Evelyn Aghemio                 638127075 02-OCT-79
```

Note: you can use the "remap_data" option in the impdp tool if you have a normal export done before  , also remember that you can use it to mask almost everything, but please take in consideration your application requirements and data integrity requirements when using it!

For more information regarding this option and to see another examples, please refer to this paper:

http://www.oracle.com/technology/products/database/utilities/pdf/datapump11g2009_transform.pdf

2) Metadata Repository and Version Control

As a DBA, I'm always looking for proactive ways to allow me to be prepared in case of a disaster strike or if an emergency release rollback is required (I love to use always the "What if" methodology), and due to these reasons, have a metadata repository and version control of it is always useful .

But how can I easily create it? Easy, first do a full backup of your database using Datapump.

```
$ expdp user/password content=metadata_only full=y directory=datapump
dumpfile=metadata_24112010.dmp
```

Note: If you want to create a repository only for objects like procedures, packages, triggers, … , all you need to do is add the parameter include=<procedures,packages,triggers,…> to your expdp command, I usually include in the dump file name the date of the dump for reference purpose and best practice.

Then use the impdp tool to create the SQL file that will allow you to create all objects in your Database. It will be something like this:

```
$ impdp user/password directory=datapump dumpfile= metadata_24112010.dmp
sqlfile=metadata_24112010.sql
```

This simple technique will allow you to create your metadata repository easily and also keep a versioning of your database objects as an extra, also if you create your repository (DB) and you want to refresh an object definition (as example let use the table emp from schema "scott"), all you will need to do is an export of the new table definition from your source database and then import it on your target database (your repository) as show bellow:

```
$ expdp user/password content=metadata_only tables=scott.emp directory=datapump dumpfile=
refresh_of_table_emp_24112010.dmp

$ impdp user/password table_exists_action=replace directory=datapump dumpfile=
refresh_of_table_name_24112010.dmp
```

3) Cloning a User

In the past when a DBA had the need to create a new user with the same structure (All objects, tablespaces quota, synonyms, grants, system privileges, etc) was a very painful experience, now all can be done very easily using Data Pump, let use as an example that you want to create the user "Z" exactly like the user "A", to achieve this goal all you will need to do is first export the schema "A" definition and then import it again saying to the Data Pump to change the schema "A" for the new schema named "Z" using the "remap_schema" parameter available with impdp.

```
$ expdp user/password schemas=A content=metadata_only directory=datapump dumpfile=
A_24112010.dmp

$ impdp user/password remap_schema=A:Z directory=datapump dumpfile= A_24112010.dmp
And your new user Z is now created like your existing user A , that easy!
```

4) <u>Creating smaller copies of production</u>

That is a very common task for a DBA, you are always having a task to create a copy of your Database (for development or test purpose) but your destination server don't have enough space to create a full copy of it! This can be easily solved with Data Pump, for this example, let say that you only have space for 70% of your production database, now to know how to proceed, we need to decide if the copy will contain metadata only (no data/rows) or if it will include the data also. Let's see how to do each way:

a) Metadata Only

First do a full export of your source database.

```
$ expdp user/password content=metadata_only full=y directory=datapump
dumpfile=metadata_24112010.dmp
```

Then, let's import the metadata and tell the Data Pump to reduce the size of extents to 70%, you can do it using the parameter "transform" available with "impdp", it represent the percentage multiplier that will be used to alter extent allocations and datafiles size.

```
$ impdp user/password transform=pctspace:70 directory=datapump
dumpfile=metadata_24112010.dmp
```

Let's do a test and see if this is really true, first let export any table of my test database (metadata only) and generate the "sql" script to see the normal size of it.

```
$expdp user/password content=metadata_only tables=user.x_integration_log_det
directory=datapump dumpfile=example_24112010.dmp

$impdp user/password content=metadata_only directory=datapump
dumpfile=example_24112010.dmp sqlfile=x_24112010.sql

CREATE TABLE "USER"."X_INTEGRATION_LOG_DET"
      ( "BATCH_NO" NUMBER(9,0),
        "SEQUENCE#" NUMBER(9,0),
        "FILENAME" VARCHAR2(200 BYTE),
```

```
        "ERROR_MESSAGE" VARCHAR2(2000 BYTE),
        "NO_OF_RECORDS" NUMBER,
        "STATUS" VARCHAR2(2000 BYTE)
      ) PCTFREE 10 PCTUSED 0 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "CUSTSERV_LOG_DET_DATA" ;
```

Above is the SQL code generated by Data Pump, you can see that the table is going to be created using 65536 for the initial extent and 1048576 for the next extent, now let's generate it again but using the transform parameter to reduce the size of it to 70% of original size.

```
$impdp user/password transform=pctspace:70 content=metadata_only directory=datapump
dumpfile=example_24112010.dmp sqlfile=x_24112010.sql

CREATE TABLE "USER"."X_INTEGRATION_LOG_DET"
    ( "BATCH_NO" NUMBER(9,0),
      "SEQUENCE#" NUMBER(9,0),
      "FILENAME" VARCHAR2(200 BYTE),
      "ERROR_MESSAGE" VARCHAR2(2000 BYTE),
      "NO_OF_RECORDS" NUMBER,
      "STATUS" VARCHAR2(2000 BYTE)
    ) PCTFREE 10 PCTUSED 0 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
STORAGE(INITIAL 45875 NEXT 734003 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "CUSTSERV_LOG_DET_DATA" ;
```

Above is the SQL code generated by Data Pump, and you can see that the table is now going to be created using 45875 for the initial extent and 734003 for the next extent, clearly reduced 30% of the original size, in other words, it works.
Please refer to Oracle documentation for more ways to use the transform parameter, you will not regret 

b) Metadata and data

First does a full export of your source database using the export parameter "sample", this parameter specify a percentage of the data rows to be sampled and unload from your source database, in this case let's use 70%.

```
$ expdp user/password sample=70 full=y directory=datapump dumpfile=expdp_70_24112010.dmp
```

Then, all you need to do as the example before is to import it telling the Data Pump to reduce the size of extents to 70%, and that's it!

```
$ impdp user/password transform=pctspace:70 directory=datapump
dumpfile=expdp_70_24112010.dmp
```

5) Creating your database in a different file structure

This is very easy to archive, all you need to do is use the parameter "remap_datafile" on your import command as the example bellow:

```
$ impdp user/password directory=datapump dumpfile=example_24112010.dmp
remap_datafile='/u01/app/oracle/oradata/datafile_01.dbf':'/u01/datafile_01.dbf'
```

6) Moving all objects from one tablespace to another

This is very easy to do it, as the previous example, all you will need to do is use the parameter "remap_tablespace" on your import command as the example bellow:

```
$ impdp user/password directory=datapump dumpfile=example_24112010.dmp
remap_tablespace=OLD_TBS:NEW_TBS
```

7) Moving a object to a different schema

All you will need to do is use the parameter remap_schema as the example bellow when importing it.

```
$ expdp user/password tables=user.table_name directory=datapump
dumpfile=table_name_24112010.dmp

$ impdp user/password directory=datapump dumpfile=table_name_24112010.dmp
remap_schema=old_schema:new_schema
```

Always remember, Data Pump is your good friend, and you will be amazed with all you can do with it to make your life as a DBA easy.

How to Make a cool magic trick

Who never had a problem with a SQL that is killing your Database Performance and you can't fix it because it's running from an external closed application that you or your developers can't touch?

Since Oracle version 10 this is a problem of the past, now you can easy solve this kind of problem using the DBMS_ADVANCED_REWRITE package, which allow you to transform/customize queries on the fly, changing for example one query with bad explain plan for another one with a good explain plan.

Before you become too excited, please remember the following restrictions:

- It does not work with bind variables.(Alternative solution at Metalink Doc ID. 392214.1)
- Only works for the SELECT statement.
- Does not work when the base table is modified through DML.

To see how it works, first we will need to grant execute privileges on the package to our user called test and allow it to create materialized views.

```
CONN sys/password AS SYSDBA
GRANT EXECUTE ON DBMS_ADVANCED_REWRITE TO test;
GRANT CREATE MATERIALIZED VIEW TO test;
```

Now let's create and populate our objects for test purpose:

```
CONN test/test

CREATE TABLE students (
  student_id               NUMBER(10),
  student_name             VARCHAR2(45),
  student_status           VARCHAR2(1),
  student_year             number(2),
  student_address          varchar2(45),
  student_city             varchar2(16),
  student_zip              number(6),
  student_social_security  number(10))
/
ALTER TABLE students
ADD CONSTRAINT pk_student PRIMARY KEY (student_id)
USING INDEX
PCTFREE 0;
BEGIN
  INSERT INTO STUDENTS VALUES (1,'PAUL COOK','Y',9,'5 Main Road','Auckland',2031,100101000);
  INSERT INTO STUDENTS VALUES (2,'HORACIO MIRANDA','Y',8,'15 Main Road','Auckland',2031,100101001);
  INSERT INTO STUDENTS VALUES (3,'SCOTT PEDERSEN','Y',7,'13 Main Road','Auckland',2031,100101002);
  INSERT INTO STUDENTS VALUES (5,'SETH PICKERING','Y',9,'12 Main Road','Auckland',2031,100101003);
  INSERT INTO STUDENTS VALUES (6,'FRANCISCO ALVAREZ','Y',11,'11 Main Road','Auckland',2031,100101004);
  INSERT INTO STUDENTS VALUES (7,'ALTMAAR VISSER','Y',9,'16 Main Road','Auckland',2031,100101005);
  INSERT INTO STUDENTS VALUES (9,'REYNALDO OCFEMIA','Y',6,'25 Main Road','Auckland',2031,100101006);
  INSERT INTO STUDENTS VALUES (15,'CAMERON PITCHES','Y',12,'31 Main Road','Auckland',2031,100101007);
  INSERT INTO STUDENTS VALUES (18,'MONIQUE GENNIP','Y',8,'71 Main Road','Auckland',2031,100101008);
  INSERT INTO STUDENTS VALUES (99,'TERRENCE LO','Y',6,'17 Main Road','Auckland',2031,100101009);
  INSERT INTO STUDENTS VALUES (100,'KIM FONG','Y',11,'16 Main Road','Auckland',2031,100101010);
  INSERT INTO STUDENTS VALUES (103,'CHRIS OPPERMAN','Y',12,'7 Main Road','Auckland',2031,100101011);
  INSERT INTO STUDENTS VALUES (104,'SCOTT TIGER','Y',6,'62 Main Road','Auckland',2031,100101012);
  INSERT INTO STUDENTS VALUES (105,'EVELYN AGHEMIO','Y',11,'32 Main Road','Auckland',2031,100101013);
  INSERT INTO STUDENTS VALUES (106,'TOMAS MUNOZ','Y',11,'18 Main Road','Auckland',2031,100101014);
  INSERT INTO STUDENTS VALUES (107,'GONZALO TORRES','Y',10,'14 Principal
Road','Auckland',2031,100101015);
  INSERT INTO STUDENTS VALUES (108,'JOHN KEY','Y',10,'12 Principal Road','Auckland',2031,100101016);
  INSERT INTO STUDENTS VALUES (109,'JOHN A','Y',7,'21 Principal Road','Auckland',2031,100101017);
  INSERT INTO STUDENTS VALUES (111,'JOHN B','Y',9,'121 Principal Road','Auckland',2031,100101018);
  INSERT INTO STUDENTS VALUES (112,'JOHN C','Y',8,'321 Principal Road','Auckland',2031,100101019);
  INSERT INTO STUDENTS VALUES (113,'JOHN D','Y',6,'35 Principal Road','Auckland',2031,100101020);
  INSERT INTO STUDENTS VALUES (114,'JOHN E','Y',12,'41 Principal Road','Auckland',2031,100101021);
  INSERT INTO STUDENTS VALUES (116,'JOHN F','Y',8,'161 Principal Road','Auckland',2031,100101022);
  INSERT INTO STUDENTS VALUES (10,'JOHN G','Y',7,'171 Principal Road','Auckland',2031,100101023);
  INSERT INTO STUDENTS VALUES (311,'JOHN H','Y',11,'353 Principal Road','Auckland',2031,100101024);
  INSERT INTO STUDENTS VALUES (312,'JOHN I','Y',7,'351 Principal Road','Auckland',2031,100101025);
  INSERT INTO STUDENTS VALUES (319,'JOHN K','Y',9,'352 Principal Road','Auckland',2031,100101026);
  INSERT INTO STUDENTS VALUES (322,'JOHN L','Y',6,'353 Principal Road','Auckland',2031,100101027);
  INSERT INTO STUDENTS VALUES (333,'JOHN M','Y',11,'354 Principal Road','Auckland',2031,100101028);
  INSERT INTO STUDENTS VALUES (343,'JOHN N','Y',6,'355 Principal Road','Auckland',2031,100101029);
  INSERT INTO STUDENTS VALUES (344,'JOHN O','Y',7,'356 Principal Road','Auckland',2031,100101030);
  INSERT INTO STUDENTS VALUES (345,'JOHN P','Y',8,'357 Principal Road','Auckland',2031,100101031);
```

```
  INSERT INTO STUDENTS VALUES (346,'JOHN Q','Y',9,'358 Principal Road','Auckland',2031,100101032);
  INSERT INTO STUDENTS VALUES (347,'JOHN R','Y',10,'359 Principal Road','Auckland',2031,100101033);
  INSERT INTO STUDENTS VALUES (350,'JOHN S','Y',11,'360 Principal Road','Auckland',2031,100101034);
  INSERT INTO STUDENTS VALUES (530,'JOHN T','Y',12,'361 Principal Road','Auckland',2031,100101035);
  INSERT INTO STUDENTS VALUES (531,'JOHN U','Y',13,'362 Principal Road','Auckland',2031,100101036);
  INSERT INTO STUDENTS VALUES (533,'JOHN V','N',6,'35 Principal Road','Auckland',2031,100101037);
  INSERT INTO STUDENTS VALUES (534,'JOHN X','N',8,'13 Principal Road','Auckland',2031,100101038);
  INSERT INTO STUDENTS VALUES (535,'JOHN Z','N',7,'135 Principal Road','Auckland',2031,100101039);
  INSERT INTO STUDENTS VALUES (536,'JOHN Y','N',11,'435 Principal Road','Auckland',2031,100101040);
  INSERT INTO STUDENTS VALUES (537,'JOHN W','Y',8,'635 Principal Road','Auckland',2031,100101041);
  INSERT INTO STUDENTS VALUES (539,'ARTUR JOHNES','Y',6,'22 Secondary
Road','Auckland',2031,100101042);
  INSERT INTO STUDENTS VALUES (540,'KING PANTHER','Y',7,'22 Secondary
Road','Auckland',2031,100101043);
  INSERT INTO STUDENTS VALUES (541,'PINK PANTHER','Y',8,'22 Secondary
Road','Auckland',2031,100101044);
  INSERT INTO STUDENTS VALUES (542,'HAROLD ROBINS','Y',9,'221 Secondary
Road','Auckland',2031,100101045);
  INSERT INTO STUDENTS VALUES (543,'CHRIS BONES','Y',8,'222 Secondary
Road','Auckland',2031,100101046);
  INSERT INTO STUDENTS VALUES (545,'TIM TOM','Y',9,'223 Secondary Road','Auckland',2031,100101047);
  INSERT INTO STUDENTS VALUES (546,'TIM JONES','Y',10,'223 Secondary Road','Auckland',2031,100101048);
  INSERT INTO STUDENTS VALUES (547,'MICHAEL JONES','Y',11,'224 Secondary
Road','Auckland',2031,100101049);
  INSERT INTO STUDENTS VALUES (548,'ANN SMITH','Y',12,'225 Secondary Road','Auckland',2031,100101050);
  INSERT INTO STUDENTS VALUES (549,'JOHN SMITH','Y',13,'226 Secondary
Road','Auckland',2031,100101051);
  INSERT INTO STUDENTS VALUES (551,'PAUL STONE','Y',6,'227 Secondary Road','Auckland',2031,100101052);
  INSERT INTO STUDENTS VALUES (552,'CARL SMITH','Y',7,'228 Secondary Road','Auckland',2031,100101053);
  INSERT INTO STUDENTS VALUES (553,'TEST','Y',8,'229 Secondary Road','Auckland',2031,100101054);
  COMMIT;
END;
/

CREATE TABLE grades (
  student_id       NUMBER(10),
  grade   NUMBER(6,2),
  grade_subject    VARCHAR2(4),
  grade_date       DATE,
  grade_note      VARCHAR2(60))
/

ALTER TABLE grades
ADD CONSTRAINT pk_GRADES PRIMARY KEY (student_id, grade, grade_subject,grade_date)
USING INDEX
PCTFREE 0;

ALTER TABLE grades
ADD CONSTRAINT fk_students
FOREIGN KEY (student_id)
REFERENCES students(student_id);

BEGIN
  INSERT INTO GRADES VALUES (553,100.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (552,95.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (551,87.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (549,87.5,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (548,90.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (547,64.7,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (546,85.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (545,88.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (543,98.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (542,95.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (541,94.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (540,94.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (539,95.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (1,88.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (2,98.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (3,98.7,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (5,96.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (6,97.0,'ENGL',sysdate,null);
  INSERT INTO GRADES VALUES (7,90.0,'ENGL',sysdate,null);
```

```
INSERT INTO GRADES VALUES (9,91.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (15,92.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (18,93.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (99,94.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (100,95.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (533,98.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (534,100.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (535,100.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (536,99.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (537,88.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (530,88.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (531,88.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (103,67.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (104,56.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (105,93.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (106,88.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (107,72.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (108,71.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (109,68.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (111,77.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (112,87.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (113,65.5,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (114,34.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (116,91.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (10,98.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (311,78.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (312,88.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (319,67.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (322,89.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (333,95.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (343,91.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (344,98.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (345,87.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (346,93.0,'ENGL',sysdate,null);
INSERT INTO GRADES VALUES (347,99.0,'ENGL',sysdate,null);
COMMIT;
INSERT INTO GRADES VALUES (553,100.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (552,95.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (551,87.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (549,87.5,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (548,90.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (547,64.7,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (546,85.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (545,88.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (543,98.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (542,95.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (541,94.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (540,94.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (539,95.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (1,88.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (2,98.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (3,98.7,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (5,96.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (6,97.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (7,90.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (9,91.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (15,92.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (18,93.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (99,94.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (100,95.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (533,98.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (534,100.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (535,100.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (536,99.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (537,88.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (530,88.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (531,88.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (103,67.0,'MATH',sysdate-7,null);
```

```
INSERT INTO GRADES VALUES (104,56.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (105,93.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (106,88.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (107,72.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (108,71.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (109,68.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (111,77.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (112,87.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (113,65.5,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (114,34.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (116,91.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (10,98.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (311,78.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (312,88.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (319,67.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (322,89.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (333,95.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (343,91.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (344,98.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (345,87.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (346,93.0,'MATH',sysdate-7,null);
INSERT INTO GRADES VALUES (347,99.0,'MATH',sysdate-7,null);
COMMIT;
INSERT INTO GRADES VALUES (553,100.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (552,95.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (551,87.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (549,87.5,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (548,90.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (547,64.7,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (546,85.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (545,88.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (543,98.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (542,95.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (541,94.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (540,94.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (539,95.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (1,88.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (2,98.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (3,98.7,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (5,96.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (6,97.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (7,90.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (9,91.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (15,92.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (18,93.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (99,94.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (100,95.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (533,98.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (534,100.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (535,100.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (536,99.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (537,88.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (530,88.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (531,88.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (103,67.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (104,56.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (105,93.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (106,88.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (107,72.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (108,71.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (109,68.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (111,77.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (112,87.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (113,65.5,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (114,34.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (116,91.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (10,98.0,'BIOL',sysdate,null);
INSERT INTO GRADES VALUES (311,78.0,'BIOL',sysdate,null);
```

```
  INSERT INTO GRADES VALUES (312,88.0,'BIOL',sysdate,null);
  INSERT INTO GRADES VALUES (319,67.0,'BIOL',sysdate,null);
  INSERT INTO GRADES VALUES (322,89.0,'BIOL',sysdate,null);
  INSERT INTO GRADES VALUES (333,95.0,'BIOL',sysdate,null);
  INSERT INTO GRADES VALUES (343,91.0,'BIOL',sysdate,null);
  INSERT INTO GRADES VALUES (344,98.0,'BIOL',sysdate,null);
  INSERT INTO GRADES VALUES (345,87.0,'BIOL',sysdate,null);
  INSERT INTO GRADES VALUES (346,93.0,'BIOL',sysdate,null);
  INSERT INTO GRADES VALUES (347,99.0,'BIOL',sysdate,null);
  COMMIT;
  INSERT INTO GRADES VALUES (553,100.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (552,95.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (551,87.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (549,87.5,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (548,90.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (547,64.7,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (546,85.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (545,88.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (543,98.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (542,95.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (541,94.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (540,94.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (539,95.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (1,88.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (2,98.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (3,98.7,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (5,96.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (6,97.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (7,90.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (9,91.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (15,92.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (18,93.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (99,94.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (100,95.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (533,98.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (534,100.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (535,100.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (536,99.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (537,88.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (530,88.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (531,88.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (103,67.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (104,56.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (105,93.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (106,88.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (107,72.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (108,71.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (109,68.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (111,77.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (112,87.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (113,65.5,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (114,34.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (116,91.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (10,98.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (311,78.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (312,88.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (319,67.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (322,89.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (333,95.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (343,91.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (344,98.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (345,87.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (346,93.0,'ARTS',sysdate,null);
  INSERT INTO GRADES VALUES (347,99.0,'ARTS',sysdate,null);
  COMMIT;
END;
/
```

Now let simulate that you found the SQL bellow running in your Database:

```
SQL> Explain plan for
select student_name,avg(a.grade) from grades a, students b
where b.student_social_security = 100101016
and   b.student_id = a.student_id
group by student_name
/



STUDENT_NAME                                AVG(A.GRADE)
------------------------------------------- ------------
JOHN KEY                                              71

SQL> SELECT * FROM TABLE(dbms_xplan.display);

Plan hash value: 3187331965
-------------------------------------------------------------------------------------------
| Id  | Operation           | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |           |     4 |   304 |     5  (20)| 00:00:01 |
|   1 |  HASH GROUP BY      |           |     4 |   304 |     5  (20)| 00:00:01 |
|   2 |   NESTED LOOPS      |           |     4 |   304 |     4   (0)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL| STUDENTS  |     1 |    50 |     3   (0)| 00:00:01 |
|*  4 |    INDEX RANGE SCAN | PK_GRADES |     4 |   104 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   3 - filter("B"."STUDENT_SOCIAL_SECURITY"=100101016)
   4 - access("B"."STUDENT_ID"="A"."STUDENT_ID")
Note
-----
   - dynamic sampling used for this statement
```

You can see that the SQL above is doing a full scan to the students table, after a few modifications we have another SQL, a little more efficient, and it will be:

```
SQL> Explain plan for
select student_name,avg(a.grade) from grades a, students b
where b.student_id = 108
and   b.student_id = a.student_id
group by student_name
/


STUDENT_NAME                                AVG(A.GRADE)
------------------------------------------- ------------
JOHN KEY                                              71

Execution Plan
----------------------------------------------------------



Plan hash value: 3300694555
-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
```

```
|   0 | SELECT STATEMENT              |            |     4 |   252 |     2   (0)| 00:00:01 |
|   1 |  HASH GROUP BY                |            |     4 |   252 |     2   (0)| 00:00:01 |
|   2 |   NESTED LOOPS                |            |     4 |   252 |     2   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID| STUDENTS   |     1 |    37 |     1   (0)| 00:00:01 |
|*  4 |     INDEX UNIQUE SCAN         | PK_STUDENT |     1 |       |     1   (0)| 00:00:01 |
|*  5 |    INDEX RANGE SCAN           | PK_GRADES  |     4 |   104 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   4 - access("B"."STUDENT_ID"=108)
   5 - access("A"."STUDENT_ID"=108)
Note
-----
   - dynamic sampling used for this statement
```

You can see that the second SQL is more efficient that the first one, and for that reason we will order Oracle to replace the bad SQL for the good one every time the bad SQL is executed, how we can do it? Easily, the magic will be:

```
SQL> ALTER SESSION SET QUERY_REWRITE_INTEGRITY = TRUSTED;
SQL> BEGIN
       sys.dbms_advanced_rewrite.declare_rewrite_equivalence (
       name              => 'test_rw1',
       source_stmt =>
       'select student_name,avg(a.grade)
       from grades a, students b
       where b.student_social_security = 100101016
       and b.student_id = a.student_id
       group by student_name',
       destination_stmt =>
       'select student_name,avg(a.grade)
       from grades a, students b
       where b.student_id = 108
       and   b.student_id = a.student_id
       group by student_name',
       validate  => false,
       rewrite_mode      => 'text_match');
     END;
     /
```

Let's now see if the magic really works:

```
SQL> Explain plan for
select student_name,avg(a.grade) from grades a, students b
where b.student_social_security = 100101016
and   b.student_id = a.student_id
group by student_name
/

STUDENT_NAME                                AVG(A.GRADE)
------------------------------------------- ------------
JOHN KEY                                              71

SQL> SELECT * FROM TABLE(dbms_xplan.display);

Execution Plan
----------------------------------------------------------

Plan hash value: 3300694555
-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |            |     4 |   252 |     2   (0)| 00:00:01 |
```

```
|   1 |  HASH GROUP BY                |            |     4 |   252 |     2   (0)| 00:00:01 |
|   2 |   NESTED LOOPS                |            |     4 |   252 |     2   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID| STUDENTS   |     1 |    37 |     1   (0)| 00:00:01 |
|*  4 |     INDEX UNIQUE SCAN         | PK_STUDENT |     1 |       |     1   (0)| 00:00:01 |
|*  5 |    INDEX RANGE SCAN           | PK_GRADES  |     4 |   104 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   4 - access("B"."STUDENT_ID"=108)
   5 - access("A"."STUDENT_ID"=108)
Note
-----
   - dynamic sampling used for this statement
```

The magic is done, now every time the source_stmt is execute it will be replaced by the destination_stmt with a better execution plan

Enjoy this amazing trick!

The 3 DBAs

To finish this paper, I'll like to explain what are in my opinion the 3 kind of DBAs we have currently available in the IT market:

- The Firefighter DBA
- The Proactive DBA
- The Balanced DBA

Now let's see what each type of DBA means for me.

a) The Firefighter

Is a DBA who is constantly fighting a fire at the time, this kind of DBA have a great knowledge in how to solve problems, but due that he or she is always busy solving problems never have enough time to do nothing else. This DBA forget what is to be proactive, and in most of the cases, the problems that keep him or she busy all the time are being caused by themselves, due that they are not taking care of the Database as required and only are fixing something when something go wrong, they use the mentality "if is not broken, don't touch it". This kind of DBA loves the adrenaline and love to be called the hero at the end of the day, but hate to be on call due to all calls received when on call!

b) The Proactive

This kind of DBA, is always looking to solve possible problems before it become a real problem, is always running checklists and setting his or her monitoring system to use proactive thresholds, this way avoiding to receive unnecessary calls during nights and weekends. Their Databases are stable most or all the time and due to this situation most of this kind of DBAs forgets what to do in case of a serious emergency situation. Another common complain this kind of DBA have, is that they do not receive any recognition from their bosses, , some were also fired or replaced due that their company think that they don't need a DBA because nothing bad never happen to their Databases.

c) The Balanced

You can easily see that the two previous stereotypes have a good side and a bad side of their behaviors, the final type of DBA, the Balanced one will take the good side of each previous stereotype, finally creating the ideal DBA.

How can this be possible?

Easily, The Balanced DBA, will behave as the proactive DBA, but will use his free time to keep learning and practicing firefighter skills. Just like my previous example regarding proactive backup early in this paper. This kind of DBA will practice backup & recovery in a test environment from any backup on tape. This exercise will allow the DBA to test and improve his/her recovery skills, test if the tapes are ok and know exactly what to do in case of a recover be necessary.

Also this kind of DBA is constantly practicing tutorials, like the ones from Oracle by Example (OBE) at OTN, to improve his/her soft skills and learn new techniques. And to avoid any possible wrong interpretation of their work, this kind of DBA is constantly involving his/her boss in what is happening, and always generating a monthly report showing:

- Current Status of Databases
- Possible problems detected during the month
- Problems avoided/Risk managed
- Resume of daily checklists
- Status of Backups
- Result of Simulations Tests (recover, OBE, etc)

With this simple report, your company will see that you really care about their data, and will understand what you are doing for them!

After learn about this 3 type of DBA in the market, can you tell what type are you? And what can you do to improve your DBA career?