2nd International Conference and workshop on Emerging Trends in Technology (ICWET) 2011
Proceedings published by International Journal of Computer Applications® (IJCA)

Enhancement of SDLC process by integrating it with SQL-PASS (SQL Performance Assurance Services)

Dattatraya M. Tendulkar
Tata Consultancy Services Limited, Mumbai, India

Chetan Phalak
Tata Consultancy Services Limited, Mumbai, India

ABSTRACT
The Software Development Life Cycle (SDLC) Models which are used in the software development process provide the control over the execution process and help to track the project progress. In this whole process, performance testing stage comes quite late and that is limited to the load testing where the system is tested against predefined number of concurrent users doing various transactions in the predefined pattern. This process gives the fair idea about the performance of the developed application but this is not adequate to test the impact of data volumes as test database is populated with the bare minimum data which is needed for the functional testing.

The data retrieving from a table having 100 rows will be definitely faster than a table having 10 millions rows. Impact of volume of data is not getting considered in the performance testing process and this holds the developer/tester back from assuring the performance in the production environment. This is very critical issue as the information systems are becoming more and more complex and it is very common these days where database sizes are in hundreds of GBs. The tools available in the market and the existing methodologies are suitable for the production environment but not effective in the development environment. This creates a gap between database application development and its deployment in the production environment. Therefore assuring the performance against high volume is an indisputable problem faced by the application developer and tester.

In this paper, we have discussed a tool 'SQL-PASS' (SQL Performance Assurance Services) based on database emulation technique and its integration with SDLC process which helps in validating and assuring the SQL performance at different stages of SDLC.

General Terms
SQL Performance

Keywords
Database Emulation, SQL Performance, Query Optimization, Explain plan, Database statistics, Extrapolation of Database statistics, query execution time forecasting

1. INTRODUCTION
Organic/inorganic growth and sometimes audit/legal requirements result into huge data volumes. In today's automation world, information requires to be made available at a 'key' press. The speed at which the information is made available is very important. The software development process models are followed to minimize the risk in the execution of project but this is not sufficient to assure the optimum performance in the production. This is happening because performance guidelines are not explicitly mentioned in the Software Development Process Models.

The other important factor is 'Time to Market', which is very crucial and any delay in it may result in to loss of business opportunities which will have the financial implications. This forces the development team to keep the focus on the validating the result rather than how they are getting those results. The code, which is written under such circumstances, may be producing the result as per the business requirement but may not be optimized which can handle the large volumes. To minimize the errors, more thrust is given on the following the standards and review process. However it cannot be full proof as it is a manual process and cannot give the guarantee of optimized performance.

Adrian Bonar has proposed a concept of 'Surrogate Database Testing' which creates a very lightweight library to provide an interface which is independent of the actual physical implementation of the underlying data source [1]. A surrogate database is a data source, such as an XML file which has same structure as backend database of the application under test. It provides a mechanism for thoroughly testing the functionality but does not address impact of high data volume.

The application developer have adopted an option of generating the synthetic data to find out the impact of data volume on performance. This helps to identify the queries which need to be optimized to perform well against high data volumes. The developers either write their own scripts or use the data generation tools to generate the data required for the testing.

DataXplod [2] is one of such tool which has been developed in TRDDC and comes with the additional feature of data masking and maintaining the referential integrity between the generated data. This is a very effective tool to test the performance before deployment or simulating the production data volumes. But this is always not possible because either low-end machine is used for the development or the development server is shared across many development projects. These low-end machines are not capable of handling high data volumes.

To handle this problem, Hornibrook [3] had invented a method of simulating the system catalogs which will be used while generating the access plan for the query (Patent US6615222 published in 2003). He defined a process for evaluating the performance of an operational database system without interfering with the normal operations of the operational database. The process involves the creation of a test database which contains the system catalogs of the operational database as user tables. The test database is used to compile queries using these simulated system catalogs. The access plan generated by the compilation in the test database can then be used to evaluate the performance of the queries. This process is using the user defined catalogs instead of original system catalogs.

Similar work was done by Chaware, Brown Douglas [4] and their team (Patent US7155428 published in 2007). Instead of using simulated catalogs, they proposed to superimpose the target system information on test system. They developed a user interface having various options that are selectable by a user for exporting the environment information from one or more target database systems. Exported data from the one or more target database systems is stored in corresponding files. The user interface of the system emulation tool also provides options to enable a user to import the environment information to a test system. In the test system, the environment information is used to create an environment that emulates the environment of the target database system.

Microsoft Research team [5] had done work on similar line and gone one step ahead in their 'AutoAdmin' physical design project where they had proposed an idea of virtual modification of the database design to find the impact on the SQL performance. The APIs were developed to manipulate the hypothetical indexes and the statistics. A 'Create Hypothetical Index' command creates metadata entry in the system catalog which defines the index. It also facilitates to generate the statistics that describes the distribution of values of the column(s) of a Hypothetical Index via the use of sampling. This facilitates the DBA to analyze the impact of a physical design change without disrupting normal database operations.

The database vendors also realized that query execution based on RBO (Rule Based Optimizer) is not sufficient as it is not considering the database volumes while deciding the execution plan. This problem has been addressed in the next generation optimizer CBO (Cost Based Optimizer) which has been enhanced to consider external factors when determining the best execution plan for SQL query. CBO takes an SQL and tries to weigh different ways (plan) to execute it. For each plan cost is attached and this cost represents the resource utilization to retrieve the data [6]. Execution cost comprises of

CPU Cost – Number of machine cycles required for query execution

I/O Cost – Number of physical block reads required for query execution

The I/O operations and CPU utilization will vary as the database volumes are changing and it will try to find out the optimum path to execute the query.

The concept of manipulating the test database statistics is well accepted within the industry and most of the databases vendors are providing the APIs to do the same. (Oracle provides DBMS_STATS package to modify the database statistics. [7][8])

To manipulate the statistics, database administration rights are required for which development or testing team has to depend upon the database administrator. Sometimes still it is not feasible to modify the statistics as it can impact the other projects who are sharing the same environment. All these problems are addressed in SQL-PASS and made it as an independent service which provides more features like auto environment creation and building production statistics from development statistics.

2. SQL-PASS

We need to have the production statistics to superimpose it on test environment. But in new development project it is not possible to have the production statistics as the project does not exist in the production environment. This constrain has been taken care in the SQL-PASS and it allows to take the statistics from the available environment and extrapolate it to match the production volume.

The other possibility is the production statistics is available but wanted to know how the SQLs will perform against futuristic volumes. SQL-PASS provides to extrapolate it further to do the what-if analysis. The extrapolation can be achieved in two ways.

I) Extrapolation by percentage

This facilitates the user to specify the percentage by which he/she wants to extrapolate the statistics and accordingly the percentage will be applied to all the tables within the schema.

II) Extrapolation by specifying number of rows

Tables within schema can be broadly classified as static tables, master tables, transaction tables and reporting tables. The ratio by which the data gets increased will not be the same for all tables. The data in transaction and reporting tables is increasing at higher speed than the master tables and there will not be much increase in static tables.

Applying the same extrapolation percentage across the schema may not be accurate as increase in data volume is not uniform across schema. SQL-PASS facilitates to specify the number of rows for selective tables. This help to extrapolate the selective tables as per the row mentioned and the other tables are extrapolated as per the percentage. This helps to match the volume statistics close to the production volume. This is very important as accuracy of the result depends upon how correctly the extrapolated volume statistics is matching with the futuristic production statistics.

The other most important problem which is addressed in SQL-PASS is the dependency on the availability of development or test environment. It is very common phenomenon where software development work is outsourced to IT vendors to reduce the project cost. This outsourced work is mostly done from the off-shore or near shore. Though outsourced team is sitting at geographically different location, they are having access to common network provided by the client for whom they are working. Though it is a development environment, vendor team is restricted with limited access and administrative rights are kept with their own team.

The process of superimposing statistics updates the system catalogs/tables and this is not feasible without administrative rights. Every time whenever vendor team wants to make the changes in statistics, they have to request the DBA. After satisfactory explanation DBA will make the changes. The procedural formalities and difference in time zone in case of off-shore development delays the approval process and impact the project schedule.

Sometimes it is not possible to change the statistics as the same environment is shared within various projects and changing statists may impact them. The updating the development/test environment is eliminated in SQL-PASS.

It allows user to upload the dump of database (system & schema) statistics along with the blank schema. The uploaded dumps and specified extrapolation parameters are used to create the emulated environment. Once the environment is ready, it can be used to check the performance of SQLs. The some of the exclusive features of SQL-PASS are as follows:

- Helps to set up the 'Emulated' Environment on the Cloud or virtual machine – Ready to use as it is not having dependency on any existing environment.
- No need to install any application on development server
- Multiple options to extrapolate the statistics
- Totally secure as each user is allowed to access his/her own project only
- No data population done in the emulated environment
- Hiring the service just in time and just for the period of need

3. SQL-PASS Business Architecture

SQL-PASS is implemented on the Linux server as a service (fig 1). The frontend is a web based application developed in Java and the backend is having oracle database and backend programs are combination of pl/sql, C and shell scripts.

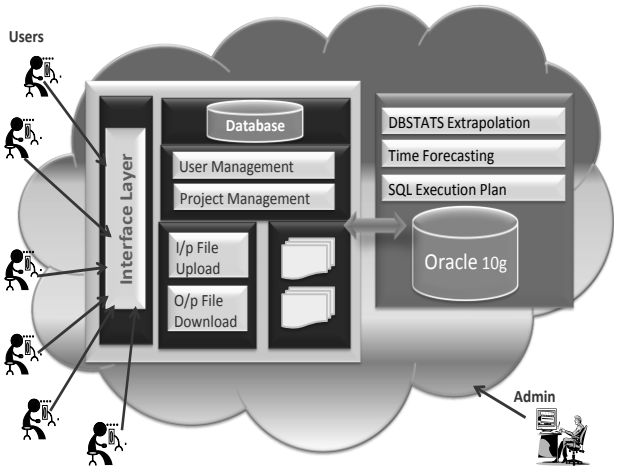

Fig. 1 SQL-PASS Business Architecture

4. Conventional SDLC Process

In conventional process, the code development starts after the high level and low level design is done. The main steps in development are shown in the fig 2..

The ETVX (Entry-Task-Validate-Exit) model based on the conventional SDLC is a well defined process which makes sure that the developed application will meet the business requirements but does not give any pointers about the performance of the application. Some of the problems surface during load testing but that is not sufficient because during the load testing concurrency of transactions is considered but most of the time database volumes are negligible as compared to the production volumes.

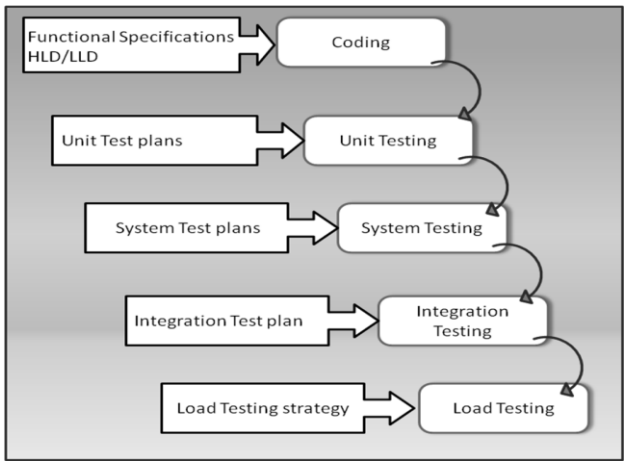

Fig. 2 Conventional SDLC Process

5. Integration of SQL-PASS with SDLC

The SQL-PASS has been integrated with the development process to deliver quality code which is functionally correct as well as the performance is assured. (fig. 3)

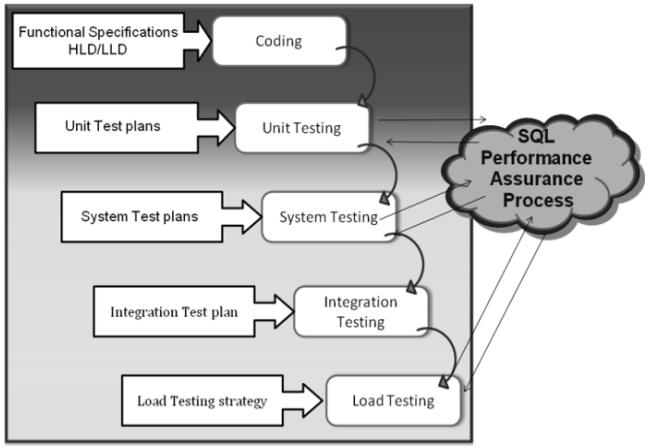

Fig. 3 SDLC with SQL-PASS

- Unit/system testing with SQL-PASS

During coding, the user can check on-line the performance of newly written SQLS using SQL-PAS.

- Load Testing with SQL-PASS

For load testing, the application is set up on production a like server to find out how the application will perform under the load condition. Testing in single user mode with SQL-PASS helps in separating out the impact of high data volume from the impact of concurrent transactions load. The problematic SQLs can be worked upon with the help of SQL-PASS to improve performance, before testing for concurrent transactions.

- Post Production Usage

In production, sometimes performance starts degrading as data volume increases. In such cases SQLs need to be tuned to get the optimum performance. During tuning, the user needs to fire modified SQLs many times to test performance. Testing SQLs in the production is risky so it is recommended to use SQL-PASS to test the performance.

6. CASE STUDY – SQL-PASS DEPLOYMENT IN THE LIVE PROJECT

A very large multi-national organization from automobile sector wanted to migrate their one of the core applications from legacy system to open platform. In this process, the application has been rewritten and data has been migrated from the legacy system to oracle 10g.

6.1 Challenges involved

This has been a huge project where multiple IT vendors are involved and responsibility has been distributed among them. The TCS project team has been responsible for the application development.

Generally when a newly developed application move to production, it handles relatively less data and the data volumes start increasing with respect to time. In such cases, one has sufficient time to monitor the system after deployment in the production and do the analysis of various transactions type to identify the probable bottlenecks. Accordingly corrective action can be planned to handle the future volumes.

Here this approach was not feasible because after deployment the application would need to handle 40GB data from the day one and expected to be double in couple of years. It was a necessity to assure the performance not only against the current volume but futuristic volumes. Writing the efficient SQLs and assuring the performance against high volume were the most difficult challenges in the project.

Alternatives available to overcome the challenges

- Generate data using tools or preparing the data generating scripts

One option was to use data generation tool to generate the enough test data to match the production volume. This would have facilitated to check the performance against high volume but it was not that feasible because of the following reasons.

- The development machine was not big enough to manage high volume

- Going for the bigger machine would have increased the project cost.

- Purchasing the tool would have increased project cost.

- Preparing the scripts to populate the data would have required extra time and efforts which in turn would have impact on the cost of the project.

- Generating the data for all the business transactions/ reports is very complex and time consuming so covering all possibilities was not feasible.

- Use migrated Data

Using migrated data was the better alternative as it would have covered all types of transactions and reports related data but it was not feasible because of the following reasons.

- The data migration activity was under control of organization's internal IT team and sharing the production data would have violated the security policy.

- The development machine was not big enough to handle production volume.

- Use masked data

Using mask data was another alternative as it would have given the access to production data and at the same time protected the secrecy of the data. This would have helped to check the performance against high volume but it was not feasible because of the following reasons.

- Data masking would have been one extra process in whole cycle.

- Purchasing the data masking tool would have increased the project cost.

- Development machine was not big enough to manage high volume

- Use SQL-PASS

Considering pros and cons of available alternatives, it was found that SQL-PASS would be the cost effective because it eliminates the need of

- Large data generation

- High power servers

- Costly tools.

The following section describes how SQL-PASS has been used to track the performance.

6.2 Procedure to use SQL-PASS

I) Capture the volume baseline from the Production Database. (In case of new application, volume baseline can be taken from the development environment.) The baseline contains

- System statistics: CPU speed, multi block read time, single block read time etc. Multi block read is the average time to read a multi-block sequentially and single block read time is the average time to read a single block randomly. These are very important system parameters which influence the query execution plan.

- Table Statistics: Number of rows, blocks, average row length etc.

- Column Statistics: Number of rows, distinct values & nulls, data distribution etc.

- Index Statistics: distinct values & nulls, density, hi-low values, average length etc.

Export the volume baseline so that it can be used for superimposing it on another environment.

II) Export schema without data which will be used to create blank schema.

III) Upload the volume baseline and schema dumps in SQL-PASS

IV) Create the Emulated environment for uploaded schema. This creates the empty schema on cloud and super imposes the supplied statistics on it.

V) Extrapolate the volume statistics to emulate the production statistics.

VI) Upload the SQLs in predefined format in Emulated environment.

VII) Process the SQLs in Emulated environment and capture the cost details and explain plan.

VIII) Forecast the SQL execution time.

IX) View online or download the generated reports to analyze the SQL performance.

6.3 Technical Benefits to the project

The integration of development process with SQL-PASS has helped the project to identify the futuristic performance problems and validate the changes after optimizing the SQLs. The emulated environment has been created by applying 700 % extrapolation factor at schema level and the key tables have been extrapolated further to match with the production volumes.

Table 1 – Extrapolated number of rows for application tables

Table Name	Number of rows
VEHICLE_EVENT_DETAILS	46875485
VEHICLE_OPTIONS	37475421
VMS_SAGA2_CLAIM_DETAILS	8768458
VEHICLE	8185785
SALES_FAMILY	6160500
CERT_OF_CONFORMITY_HEADER	4237404
ORDER	4086100
CERT_OF_CONFORMITY_RN	3898574
SALES_FAMILY_ELEMENT	2465789
CERT_OF_CONFORMITY_R2	1868584
PNR_DETAILS	1276895
MODEL_PACKAGE_PNR	606065

Most important SQLs (count 110) have been tested with SQL-PASS. It has helped to identify the SQLs which need to be optimized.

Table 2 - Result provided by SQL-PASS

Query Reference	Development		Emulated	
	Cost	Time second	Cost	Time second
/*+ PERF-15-JUL-2009-10-30-23-0094*/	134	0.05	1586174	236.74
/*+ PERF-15-JUL-2009-10-30-23-0110*/	124	0.07	1556076	351.37
/*+ PERF-15-JUL-2009-10-30-23-0073*/	563	1.09	179521	208.54
/*+ PERF-15-JUL-2009-10-30-23-0070*/	563	2.99	169536	540.23
/*+ PERF-15-JUL-2009-10-30-23-0099*/	74	0.05	54223	21.98
/*+ PERF-15-JUL-2009-10-30-23-0100*/	84	0.02	54223	7.75
/*+ PERF-15-JUL-2009-10-30-23-0032*/	11	0.19	42559	441.07
/*+ PERF-15-JUL-2009-10-30-23-0096*/	84	0.59	39730	167.43
/*+ PERF-15-JUL-2009-10-30-23-0105*/	35	0.06	12598	14.04
/*+ PERF-15-JUL-2009-10-30-23-0120*/	142	0.98	7179	32.2
/*+ PERF-15-JUL-2009-10-30-23-0098*/	4	0.05	4201	34.13
/*+ PERF-15-JUL-2009-10-30-23-0069*/	12	0.47	3565	90.76
/*+ PERF-15-JUL-2009-10-30-23-0118*/	189	0.08	2208	0.61

/*+ PERF-15-JUL-2009-10-30-23-0103*/	12	0.04	1792	3.88

The cost and execution time of SQLs has been brought down by applying the following measures.

- Adding new indexes

New indexes have been created to improve the performance.

Table 3 – Recommended indexes to improve the performance

TABLE_NAME	Suggested Index
BUSINESS_PARTNER	SENT_TO_KUBA
DATALOADER_ERROR_FILES_DET	ERROR_FILE_STATUS
DATALOADER_GRP_INTERFACE_MAP	GROUP_ID
DATALOADER_PROCESS_FILE_DET	INTERFACE_ID
VEHICLE_EVENT_DETAILS	EVENT_ID
VEHICLE_POLICIES	DATA_RECIEVED_DATE
VEHICLE_POLICY_OWNER_DETAILS	SENT_TO_KUBA
VEHICLE_AUDIT_HISTORY	AUDIT_EVENT_TYPE_ID
VEHICLE	REGISTRATION_NUMBER

- Removing the unwanted conditions

The conditions which are the supper set of other conditions have been removed safely as it was not impacting the business logic.

Example :

Audit_Event_Timestamp IS NOT NULL AND

Trunc(Audit_Event_Timestamp) >= To_date('01-May-09', 'DD-MON-YY') AND

Trunc(Audit_Event_Timestamp) < To_date('01-Jun-10', 'DD-MON-YY')

In this example, 'Not Null' condition can be removed without impacting business functionality.

- Adding appropriate conditions

During analysis, it has been found that in some SQLs same table has been used in main query as well as the sub-query but filtration conditions have been mentioned only in sub-query. Adding appropriate filter conditions in main query have improved the data retrieval process and cost has come down.

In following example the execution of main query has been totally dependent on sub-query i.e. each record form main query was checked with sub-query result set. The additional condition has been added in main query to filter the records before checking with sub-query result set.

Query	/*+ PERF-10-JUL-2009-06-48-56-0001*/ SELECT EVENT_ID,EVENT_DATE,VIN, MILEAGE FROM T_VEHICLE_EVENT_DETAILS WHERE EVENT_DATE = (SELECT MAX(EVENT_DATE) FROM T_VEHICLE_EVENT_DETAILS WHERE VIN='TMBCS21Z292043529' AND EVENT_ID='OWN')				
Level	Operation : Options	Object Name	Cost	IO COST	CARDINALITY
1	SELECT STATEMENT : -		1586174	1584048	7
2	-TABLE ACCESS : FULL	T_VEHICLE_EVENT_DETAILS	1555959	1553853	7
3	-SORT : AGGREGATE		1	1	1
4	-TABLE ACCESS : BY INDEX ROWID	T_VEHICLE_EVENT_DETAILS	30215	30195	963419
5	-INDEX : RANGE SCAN	T_VEHICLE_EVENT_DETAILS_U	25173	25163	2214585

Query	/*+ PERF-10-JUL-2009-06-48-56-0002*/ SELECT EVENT_ID,EVENT_DATE,VIN, MILEAGE FROM T_VEHICLE_EVENT_DETAILS WHERE VIN='TMBCS21Z292043529' and EVENT_DATE = (SELECT MAX(EVENT_DATE) FROM T_VEHICLE_EVENT_DETAILS WHERE VIN='TMBCS21Z292043529' AND EVENT_ID='OWN')				
Level	Operation : Options	Object Name	Cost	IO COST	CARDINALITY
1	SELECT STATEMENT : -		55391	55359	1
2	-TABLE ACCESS : BY INDEX ROWID	T_VEHICLE_EVENT_DETAILS	25176	25164	1
3	-INDEX : RANGE SCAN	T_VEHICLE_EVENT_DETAILS_U	25175	25163	1
4	-SORT : AGGREGATE		1	1	1
5	-TABLE ACCESS : BY INDEX ROWID	T_VEHICLE_EVENT_DETAILS	30215	30195	963419
6	-INDEX : RANGE SCAN	T_VEHICLE_EVENT_DETAILS_U	25173	25163	2214585

Fig. 4 Example SQL before and after optimization

- Re-writing the conditions to use the indexes

Some of the conditions have been re-written to force the optimizer to use the indexes.

- Table Partitioning

Some of the transaction tables are having large data and expected to grow in future so it has been recommended to have table partitioning to improve the performance.

SQL-PASS has enabled the proactive and helped in assuring the performance of application against the production volume.

7. What-If Analysis with SQL-PASS

One of the major international broadcast companies was expecting increase in data volumes by hundred percent. They wanted to find out the impact of increase in data volume on the performance of their daily batch jobs (46 reports). These reports were using the complex SQLs based on oracle VIEWS and their sizes were in the range of 1000 to 5000 lines. Making changes in such SQL was risky so they wanted to confirm how the SQL will perform in future when the volume will be increased. The schema was having more than 1000 tables. The number of expected rows were set for the critical tables and percentage was specified at global level which was applicable for the remaining tables to extrapolated the statistics.

The cost details and explain-plans were compared with the production environment to identify the differences. It helped to separate out the SQL whose performance would be deteriorated with increase in volume. Based on the SQL-PASS result, the following actions were taken.

- Additional indexes were defined to improve the performance
- Data partition was recommended in huge tables to handle the data in parallel.
- Archival strategy was formed to reduce the burden on the production database.

8. CONCLUSION

This document has highlighted not only the need of checking the performance against high volume at development stage but also demonstrates how SQL-PASS can help to validate the performance to minimize the post development performance problems.

9. ACKNOWLEDGEMENT

I am thankful to Dr. Rajesh Mansharamani for his support and guidance.

10. REFERENCES

[1] The Design of a System for Testing Database-Centric Software Applications Using Database Surrogates by Adrian Bonar - http://ieeexplore.ieee.org/Xplore/login.jsp?url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F5070574%2F5070575%2F05070716.pdf%3Farnumber%3D5070716&authDecision=-203

[2] High Utility Data Generation Using DataXplod - http://www.tcs-trddc.com/trddc_website/pdf/SRL/banahatti_hudgud_tactics.pdf

[3] System & Process for evaluating the performance of a database system – Publication number US 2003/0115212 A1 (http://ip.com/patent/US6615222 Inventors - Hornibrook, John F,Totonto)

[4] Chaware; Jeetendra (Andhra Pradesh, IN), Brown; Douglas P. (Rancho Santa Fe, CA), Sinclair; Paul L. (Manhattan Beach, CA), Julien; Thomas P. (San Diego, CA)

http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2Fsearch-adv.htm&r=48&f=G&l=50&d=PALL&S1=6430556.UREF.&OS=ref/6430556&RS=REF/6430556

[5] Paper "Self-Tuning Database Systems: A Decade of Progress" published by Surajit Chaudhuri & Vivek Narasayya from Microsoft Research Group

[6] Explanation of Cost-based Optimizer and Rule-based Optimizer http://members.fortunecity.com/dpafumi/CBO_and_RBO.htm

[7] Oracle APIs (package) for manipulating the database statistics http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14258/d_stats.htm

[8] Cost Control: Inside the Oracle Optimizer http://www.dba-oracle.com/art_otn_cbo.htm