Mercury Consulting Ltd.
Bahnhofstrasse 7, FL-9494 Schaan, Tel. +423 236 44 06 Fax +423 236 44 07
e-mail: office@Mercury-Consulting-Ltd.com
http://www.Mercury-Consulting-Ltd.com

# The Danger of Invalid Database Objects

*Why they are dangerous.*

*An awareness Paper for IT-Managers.*


*http://www.IT-Checklists.com*

Published in June 2005



## Conventions

The following typographical conventions are used in this book:

`Constant width`
: Used for source code

**`Constant width bold`**
: user input

[ ]
: optional elements

{ }
: mandatory choice

|
: separates choices

# Table of Contents

# Summary

This document shows the risks when operating a database with invalid objects. This document is written for members of application support operations managers. The ***operations managers*** should understand the ***risk associated when operating databases with invalid objects*** and he should solve the dispute between application support and DBA's who both argument that this is not "his" but "the others" problem.

An invalid database object (Synonym, Package, Procedure, View,..) could indicate a problem, but it might be also harmless. Nevertheless operating a database with "harmless" invalid objects makes the detection of a new invalid object indicating a serious problem impossible or much more difficult. Combined with missing controls in other areas severe malfunction might not be detected. In case of problems the troubleshooting process might require more time because investigators might also research all invalid objects.

Effort for an initial cleaning (getting rid of invalid objects) of all databases pays off, as all involved persons get awareness about risks and importance so that new systems and changes will not introduce new invalid objects. Thus you will increase the maturity level of your IT.

# Audience

- Operations Manager
- Testing Manager
- Environment Manager
- Application Support Staff
- System-DBA, Application-DBA, Development-DBA
- Quality Assurance Manager

# Introduction

In an Oracle Database an object (Synonym, Package, Procedure, View, ...) becomes invalid if one of its underlying objects is changed. If the change on the base object does not cause an error in the referencing object, a simple recompiling will make the object valid again. This recompile is automatically triggered if an invalid object is accessed. But if the change on the underlying, referenced object causes an error in the calling object, this calling object CANNOT be recompiled, and all programs calling this now invalid objects will fail.

| There are two types of invalid objects:<br><br>1) Objects which become valid when recompiling them<br>2) Objects which cannot be successfully recompiled |
|---|

Invalid objects of type 1) are not critical but they make monitoring and troubleshooting more difficult. Therefore it should be investigated why they became invalid (and implement a mechanism to recompile them after this event). As objects should be recompiled at next access those objects might be obsolete if they stay invalid for a long time. Requesting a patch (or at least written approval) from application vendor to remove those obsolete objects is the most effective way to get the database clean.

Invalid objects of type 2) could be either obsolete objects (e.g. a new version removed a table or procedure but not the public synonym) or indicate a problem.

Before drilling down to details the next chapter "Impact Analysis" should demonstrate the risks and issues caused by invalid objects.

# Impact Analysis

If you do have invalid objects in your database, then answer following questions:

| ***Question*** | |
|---|---|
| 1) How do you monitor your system for invalid objects ? | A) Our monitor system shows permanent a red flag for "invalid objects" but we ignore it because we know that we have invalid objects. We are aware that we would not detect a new invalid object.<br>B) We disabled this control in our monitoring system because we know that we have invalid objects. We are aware that we would not detect a new invalid object.<br>C) We customized our monitoring system to ignore the known invalid objects, but we would get an alarm if there is a new invalid object.<br>D) Our monitoring system does not this control on invalid objects.<br>E) We don't have database monitoring at all. |
| 2) In case of implementing a change on that database we... | ..A) detect new invalid objects because we compare during the post implementation check the invalid objects with the list of documented (known) invalid objects.<br>.. B) will not detect a new invalid object because we don't check for invalid objects at all.<br>.. C) will not detect a new invalid objects because we don't know in detail which objects have been invalid before applying the change. |

Answers to question 1:

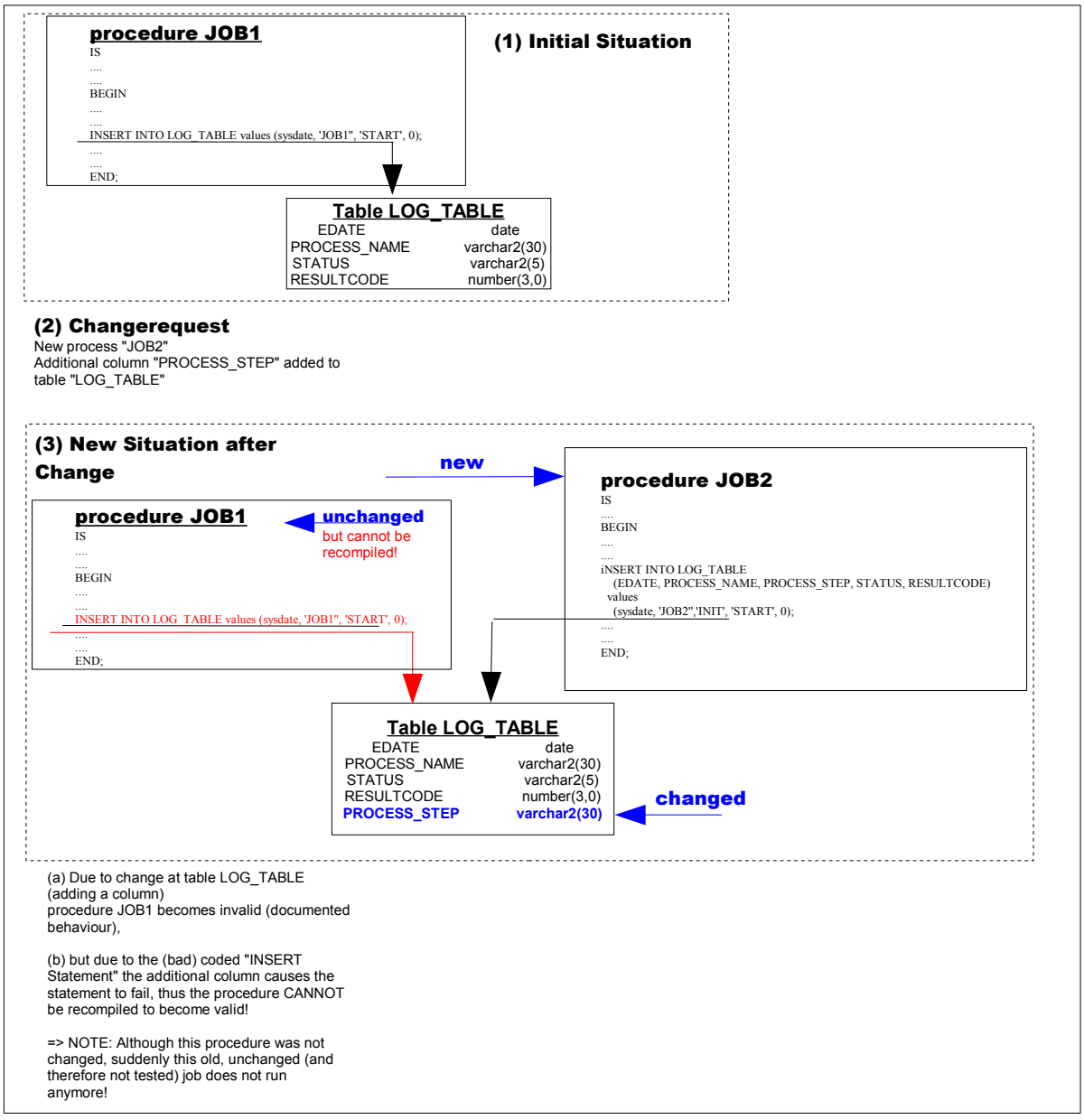
| | |
|---|---|
| C | You did a great job, but you surely know which costs the customization did cost. |
| A, B, D, E | We hope that you have excellent controls on many other places to detect malfunctions. |

Answer to question 2:

| | |
|---|---|
| A) | You are doing a great job, but you are aware about the additional effort compared to a clean (free of invalid objects) database. |
| B), C) | You hopefully have a good post implementation check to identify that after implementing this change something does not work. Take care! - The NEW module you installed or changed might work, but ANOTHER, UNCHANGED module might suddenly fail! |

# Real world case study

You should not spend too much time on discussing the different arguments in table "Blamestorming" - just take this unlucky incident as warning to keep your system (not only the database) as clean as possible and use each slightest indication of a problem as warning and investigate it.



## *The Blamestorming*

| *Blame* | *Response* |
|---|---|
| It's a development problem. You never should code an insert-statement without explicitly listing the column-names. | The developer arguments that no coding standard. And there was no code-review. |
| The testing team failed, because it should have detected the invalid procedure "JOB1" already on the testing environment. | a) Testing team did not look at all<br>b) There are such many invalid objects that nobody can identify a new invalid object.<br>c) This could not be detected, as the new change changing the LOG_TABLE and introducing procedure JOB2 was tested on test environment T2, and procedure JOB1 was installed only on test environment T1 but NOT on T2; therefore JOB1 could not become invalid. But that is a problem of "environment maintenance". |
| The testing team failed because it did not a complete re-testing of all modules | We don't have enough time to re-test all applications after each small change.<br>There is no documentation like a "Function / Entity" Matrix showing the dependencies, so we cannot identify dependent modules. |
| The post release check failed, because the invalid object was not identified and investigated after implementing the change. | The System DBA arguments that database objects in the application schema are under the responsibility of the application support team.<br>Application Support members argument that they are not database experts and don't understand what an invalid database object is.<br>Fact: There is a problem in the release process and in the CLEAR, unambiguously assignment of responsibilities. To address this quite typical problem Mercury Consulting Ltd. created **product "DBA and Application Support: Job Description and Self-Assessment".** |
| The problem that the old process JOB1 did start but immediately fail with error message added to logfile for several days is a mistake of application support, because there was neither a monitoring for errors in the log file nor an independent monitoring of successful job completion. | This issue is addressed in Mercury Consulting Ltd.'s free whitepaper "**The importance of application level monitoring**" and in the product "**Non functional Requirements"**, a requirements template with 150 non functional requirements, among those many about application level monitoring. |

Conclusion:

(1) Big problems are very seldom the result of one single mistake; therefore you should try to eliminate as many small mistakes as possible.

(2) The problem could have been identified in many places - take each chance to detect an anomaly. Keeping your systems clean will help you to detect new anomalies.

# Important Housekeeping: Clean the database from invalid objects

| *Statement* | *Comments* |
|---|---|
| We have no time | You definitely need time to document all invalid objects with an explanation why there is no problem (including explanations from Vendors or developers).<br>If you have no time to write this documentation now, you can for sure plan that time as follow-up action after the next external IS audit.<br>And you will need additional time to modify your monitoring system to ignore exactly those known and documented invalid objects.<br>Disabling the existing monitor event is about the same as removing the red light bulb in your car's dashboard indicating a problem in the break system or oil level.<br>And the author has absolutely no idea how to explain an external auditor why an standard alarm was disabled. |
| We don't know whom to ask because we don't know if those invalid objects are part of the application delivery or were created by someone else. | 1) You definitely have a problem in your change management- and configuration management process. And you should fix that before the next external IS audit.<br>2) Check the vendor's documentation |

# Even if you have no time for "cleaning" databases from invalid objects, then you must ...

The author understands that in many cases the existence of invalid objects is not the DBA's fault and not the application support member's fault, they have been introduced by others.

But nevertheless application support (if object is in the application schema), System-DBA (if object is in one of the system-schemas as "SYS", "SYSTEM", "OUTLN", ...) must document this problem and all steps done to mitigate it.

| *Nr.* | *Action* |
|---|---|
| 1) | Immediately add a "Problem" to your "Problem Management System". (This is the ITIL[1] term, in your company you might use another term for the system documenting, tracking and managing problems). |
| 2) | Document all steps which you tried to solve the problem. If it's a 3rd party application you should at least open a Anomaly, TAR, Problem at the vendor's support. The vendor might provide a patch (script) to remove the obsolete objects, or approve the removal or he will tell you that this is OK. Although the last response is not acceptable, you at least will have an official statement from the application vendor which you can show to the auditor who might list the invalid objects as defect. |
| 3a) | If you got rid of invalid objects, then you can close the problem. |
| 3b) | If you could not, but got official statements from the application vendor, then you could change the status to "Known Problem".<br>But additionally you must customize the monitoring system to ignore the DOCUMENTED, KNOWN invalid objects but to insure to detect new invalid objects. |

# Prevention for new problems of this type

As you have enough problems in living with your existing "dirty" databases you should definitely avoid that new dirty databases will be introduced. Therefore you need to add the requirement that the delivered system or change must be free of any invalid object as mandatory requirement to your requirements list or acceptance criteria.

If you do have this requirement documented but those requirements are not communicated to vendors and developers you do have a problem in the change management process.

If you use our requirements template "Non Functional Requirements", then this requirement is one of 150 other requirements.

1 ITIL® = "IT Infrastructure Library" is not an official standard but the worldwide most accepted collection of best practices for Service Delivery and Service Support. ITIL is a registered trademark of UK's OGC , Office of Government Commerce.

# How Mercury Consulting Ltd. can help you

Mercury Consulting Limited (MCL) is a professional consultancy providing experience, support and training in IS/IT operations for companies during high-time-pressure startup phase and following consolidation phase, especially in the Telecom-market.

## *Our Products*

you can purchase online and immediately download at our eBook-Shop

Our Checklists and Templates will help you to ensure that good or best practice is not only known but consistently applied!

## *Database Independent Products*

| ***Product*** | ***Benefit*** |
|---|---|
| 150 Non functional Requirements. | Requirements Template with 150 non functional requirements for selecting or developing robust, reliable, stable and manageable applications to meet the Service Level Agreements (SLA's).<br>For external RFP's (Request for Proposal) and for internal development. |
| Checklist for Data Migration | 65 important questions to identify and address or exclude typical migration pitfalls in an early phase of the project, thus ensuring the confidence for keeping the time plan. |
| Template: Systems- and Operations Handbook | Template to establish that documentation auditors like to see! |
| Interface Checklist | Those questions which you need to ask before starting the development! Requirements, Checklist and Template for Planning, Defining and Documenting Application Interfaces. |
| Checklist for Production Release and System Handover | Checklist for small and medium projects focusing on non-functional aspects for operations team.<br>Simple but effective! |
| Business Requirements for Archiving and Purging | Template with Business Requirements for Archiving & Restore & Purging. Not removing old customer data can cause conflicts with privacy laws. ***Business must act*** and clearly specify what to purge and what to archive! |
| Application Health Check: Stability Assessment | Using this template to check your systems - and DOCUMENT the findings might show you even more potential issues beyond invalid objects. |
| Technology Selection for Disaster Recovery (DR) | Standby database using transfer of transaction log / archive logs or replicating the database files ?<br>Host-based, Storage based or Switch/Fabric based replication ?<br>This document evaluates the different techologies and their advantages and constraints from an operational point of view. |
| Backup SLA / OLA: Operations Level Agreement with the Backup-Team | A template which supports the creation of an operational level agreement (OLA) between application support, database administrators and backup team. |

## *Database Specific Products*

| ***Product*** | ***Benefit*** |
|---|---|
| Template for Database Operations Level Agreement (OLA) / SLA | If a Service Level Manager needs to offer to Business a Service Level Agreement (SLA) for an End-to-End IS/IT-Service, he sign this SLA only after he arranged within IS/IT for each system or component used to provide this service an **Operations Level Agreement (OLA)** with the providing team or department.<br><br>This document provides a template for such an Operations Level Agreement (OLA) for Databases containing the agreed values and for QA-purpose also the measures implemented to reduce the likelihood of violations of those agreed values.<br><br>It does not only deal with availability, but contains also comprehensive service catalogue of advanced DBA services and a template for the Service Level Reporting (SLR). |
| Database Health Check -<br><br>Part 1: Stability Assessment | Stability Assessment of your Database.<br><br>Most Application- and Database-Crashes can be avoided when detecting early indicators and reacting to them.<br><br>***Be Proactive - Check Now!*** |
| DBA and Application Support:<br><br>Job Description and Self Assessment | Checklist to ensure that all 60 DBA-duties are assigned:<br><br>System DBA, Development DBA and Application DBA versus Application Support<br><br>If you have just a job role "DBA", but not a dedicated job role "Application DBA" those 19 duties must be explicitly assigned to either "Application Support" or to the "DBA" - otherwise they might not be executed!<br><br>***Detect unassigned tasks before an auditor reports them !***<br><br>This product addresses disputes between System DBA and Application Support (or, if existing, dedicated Application DBA) regarding the responsibility for the application's database objects. |

## *Our free Whitepapers*

http://www.mercury-consulting-ltd.com/wp/whitepapers.html

| *Whitepaper* | *Benefit and Description* |
|---|---|
| The Importance of Application Level Monitoring. | Keeping your applications free from invalid objects is an important task, but does not guarantee error free operations. This free whitepaper explains the difference between "Application Level Monitoring" to "Database Monitoring" and "System/Server/OS"-Monitoring. |
| Important Facts about Redolog- and Archivelog / Transaction Logs in Databases for Change Managers and Application Support Staff | Understanding the topic “archivelog volume” can avoid unexpected troubles when applying changes or conducting upgrades.<br>This awareness paper explains why Change Managers must ask questions about the archivelog volume created during changes and upgrades and why application support staff – and not only the DBA - must understand this topic. |
| The Danger of Invalid Database Objects | An awareness paper for Operations Managers and Application Support describing the problems and potential risks caused by invalid objects in Databases. |
| Private or Public Synonyms<br>– or no Synonyms at all ? | A decision support paper visualizing the pro's and contras on a single page in tabular form and evaluating the arguments. |
| Daily, weekly or monthly partitions? | This document evaluates the different factors impacting the decision if daily, weekly or monthly partitions should be used. Legal- , performance- and maintenance aspects are evaluated. |

## *Our Services*

http://www.Mercury-Consulting-Ltd.com/services.htm

## *About the Author*

The Author of this white paper is an

ORACLE Certified Professional

ORACLE 10*g* Certified Professional

With 12 years experience as DBA and 6 years experience in Telecommunications Companies.