國立交通大學

電機資訊學院 資訊學程

碩 士 論 文

應用 RB 與 CBR
建構一個解答擷取系統

Building a Solution-Retrieval System
Based on RB and CBR Approaches

研 究 生：李宗平

指導教授：曾憲雄 博士

中 華 民 國 九 十 五 年 六 月

應用 RB 與 CBR 建構一個解答擷取系統

Building a Solution-Retrieval System Based on RB and CBR Approaches

研 究 生：李宗平　　Student：Tsung-Ping Lee

指導教授：曾憲雄　　Advisor：Shian-Shyong Tseng

國 立 交 通 大 學

電機學院與資訊學院專班 資訊學程

碩 士 論 文

A Thesis

Submitted to Degree Program of Electrical Engineering and Computer Science

College of Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年六月

應用 RB 與 CBR 建構
一個解答擷取系統

研究生：李宗平　　指導教授：曾憲雄 博士

國立交通大學電機資訊學院 資訊學程

摘要

由於網路快速發展，服務多人上線的系統，已從傳統的兩層式 (client-server) 架構轉型成三層式 (client-application server-database)架構，因此，問題的偵測跟解問題的方法，對於這樣的領域來說，其複雜度就不斷增加。就我們所知，很多公司需要花很多人力物力去維護這樣的系統，然而公司的專家不可能一直在公司解問題，既使用文件搜尋來找答案，也可能找不到解答，所以提出根據專家系統的方法，來建構一個解答擷取系統。在本篇論文中，我們提出 Solution-Retrieval System (SRS) 架構來輔助使用者解決問題。在 SRS 中，結合 Rule Base (RB)跟 Case Based Reasoning (CBR)方法，我們使用 RB 推論來縮小問題的範圍，用 CBR 來找問題的解答，這樣處理的方式是模仿專家的處理模式，依著系統的特性，用推論的方式來找出錯誤類型，再將推論出來的錯誤類型，用專家的經驗法則，比較相似度，將相似度高的解決方案應用到問題上去解系統問題。再者，為了提高解問題的及時性，我們的系統也可以在 PDA 等小型裝置上及時的運用來解問題。未來，這樣的架構可以應用在相關的領域上，例如: IC 設計，整合供應鍊 等。

關鍵字：解答擷取、專家系統、Rule Base、Case-Based Reasoning

Building a Solution-Retrieval System Based on RB and CBR Approaches

Student: Tsung-Ping Lee Advisor: Dr. Shian-Shyong Tseng

Department of Master Program of Computer Science
National Chiao Tung University

Abstract

Due to the fast development of web services, most of the service activities have been moved from 2 tiers (client-server) to 3 tiers (client-application server-database); hence, the importance of problem diagnosis and solution retrieving in integrated domains becomes more complicated. As we know, many of companies devote lots of time and effect to deal with this problem. However, since experts are not always available, using traditional approach or knowledge management center to search solutions may still fail. Hence, the idea of developing a solution-retrieval system based upon expert system approach is proposed. In the thesis, we propose an architecture based on rule base (RB) and case-based reasoning (CBR) and build a Solution-Retrieval System (SRS) to help users to solve the problems, in which RB is used to reduce error scope and CBR is used to find the corresponding solutions. Similar to the expert's diagnosis approach, we use the SRS to diagnose the error type by RB inference, and retrieve solutions by CBR. Finally, the retrieved high similarity solution cases can be used to solve problems. Furthermore, the PDA and hand-held devices could be used in our system for solving problem promptly. In the near future, this architecture will be adapted on other related domains, e.g., IC design and Supply Chain.

Keywords: Solution-Retrieval, Expert System, Rule Base, Case-Based Reasoning

誌謝

能夠完成這篇論文，要感謝的人很多，首要感謝我的指導教授 曾憲雄 教授，教授做事的態度，研究的方法與待人接物方面，對我的影響既深且遠，能夠在教授的指導下完成學業，是一種福氣，十分感謝。同時也要感謝我的口試委員 林妙聰 教授，洪宗貝教授與 李允中 教授，他們對於這篇論文提供了不少寶貴的建議。

再來我要感謝實驗室的學長，翁瑞鋒學長、蘇俊銘學長與 王慶堯學長，除了在專家系統上專業知識的指點，對於論文也提供了不少寶貴的建議。尤其是 瑞鋒學長對於論文的協助，不宜餘力，與我度過無數個長夜，嘔心瀝血完成這篇論文。這一切都是深受曾憲雄 教授平時的教誨，所養成的良好實驗室風氣。

另外我要感謝，我的同學：碧如、亮中、秀怡、薰尹、書源、廼仁、鳳琴、仁杰，平常時對於我的協助，以及在功課方面的幫忙，讓我可以順利完成這篇論文，非常感謝。

最後我要感謝我的家人，我的女友 秋詩，在我意志薄弱的時候，給我關懷，給我鼓勵，讓我有信心可以站起來，往前走，順利完成這篇論文。要感謝的人很多，無法一一描述，謹在此獻上我最誠摯的感謝。

Content

List of Definitions

List of Examples

List of Figures

Chapter 1. Introduction

Due to the fast development of web services, most of the service activities have been moved from 2 tiers (client-server) to 3 tiers (client-application server-database), and the importance of problem diagnosis and solution retrieving in application server and database integration domain becomes important increasingly. As we know, many of companies devote lots of time and effect to deal with this problem in recent years. However, since the cost is too high and experts are not always available, using traditional knowledge management (KM) center search approaches may fail. However, the mission is critical and is difficult for novice to handle this job; once the abnormal situation happens, on-duty employees have to fix the problem as soon as possible. Therefore, the idea of developing a solution-retrieval system based on expert system approaches is proposed in this thesis.

To deal with these problems, the novices usually find the solutions from KM center, but somehow they have to face two main challenges listed below.

(1) **Problem diagnosis issue:** Since it is difficult to diagnose system for novices, diagnosis system problems have to refer other related logs and system status to diagnose precisely. Besides, it is difficult to determine which part of the log content is meaningful.

(2) **Solution retrieving issue:** Although there are traditional approaches, e.g., KM Center, novices may not know how to input the right keywords for solution retrieving.

As on-duty employees may fail to fix problems, usually they call experts for help. Then experts will reason the solutions of the problem based on their experience. Based on the problem diagnosis and solution retrieving mechanisms, the Solution-Retrieval System (SRS) was produced. We use rule based inference to infer the error type for reducing the error scope,

which imitates experts and use case based reasoning (CBR) to reason the similarity between error type and cases in case base to find out the corresponding solutions.

The system SRS includes three main components: preprocessor, inference module [22], and reasoning module. When the monitored system (Application Server and Database) is abnormal, the preprocessor retrieve error logs and system information. Then inference module infers the error type immediately. And then users have to input related keywords to trigger reasoning module to reason the solutions immediately. Therefore, the main contribution of this thesis includes, providing the hybrid architecture to solve system problems, using RB to reduce error scope, and using CBR to reason the solutions.

In this thesis, we face the challenges listed as follows.

(1) Knowledge Acquisition (KA): Using rule-based inference in KA, there are Oracle consultants and database administrators (DBA) in our company; hence KA related work and the perfection of the rules could be easily done.

(2) Similarity Calculation: Since it is difficult to define attributes in query and the existing cases, to calculate the similarity between query and cases is difficult.

This rest of thesis is organized as follows. Chapter 2 introduces the related works. Chapter 3 describes domain analysis, knowledge representation, SRS system architecture overview, and preprocessor in SRS. Chapter 4 describes System Diagnosis by Rule Base. Chapter 5 describes Solution Retrieving by CBR. Chapter 6 is the experiment processing and evaluation including system design and implementation. Chapter 7 gives future work and conclusion of the thesis.

Chapter 2. Related Work

As we know, more and more complex systems (e.g., Oracle Application Server and Oracle database) are revised in product version; it results in more and more difficult problems in trouble shooting occurred. In our study, many traditional approaches, e.g., search engine or Knowledge Management (KM) center for trouble shooting may fail; hence, we want to propose new approaches to do problem diagnosis and solution retrieving.

In order to solve the issues mentioned above, we design an architecture SRS, which can not only find the root-cause efficiently but also reuse and retain expert's experience to help users to deal with problems promptly.

2.1. Problem Diagnosis

As for problem diagnosis aspect, using probabilistic reasoning techniques can be combined with information-theoretic approach to focus on distributed system problem diagnosis [14]. This approach may be efficient in problem diagnosis by adapting multi-layers system architecture and probabilistic reasoning techniques, but lacks the corresponding solutions. Using a new data-mining algorithm with ontology-based approach to fault diagnosis [7] still lacks the solutions to aid users deal with problems. On the other related approaches, using neural network and spectral analysis to detect the operating machine fault [6] can only provide the diagnosis result for one specific domain only. A hybrid fault diagnosis expert system based on knowledge and neural network in a steel factory [16] did not provide the solutions. Some other related researches use multi-CBR based recommendation engine for e-commerce may not be efficient [9], due to lack of decision making mechanism. Hence, we propose an inference approach (Rule-Based Inference) to reduce time in problems

diagnosis and make the SRS more efficient.

As we know, problem diagnosis is difficult for IT employees due to the lack of expert's inference skills, as shown in Figure 2.1.

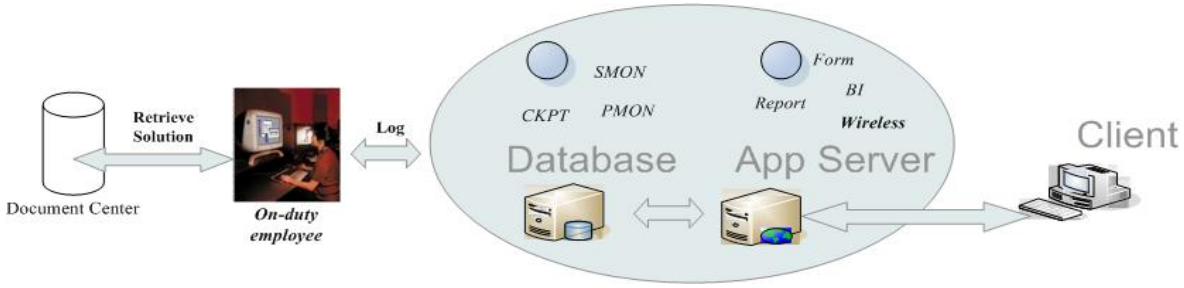


Figure 2.1. Problem diagnosis by on-duty employees

The overall architecture for trouble shooting is given, where Oracle Application Server and Oracle database server are particularly presented in Figure 2.2 and Figure 2.3, respectively. In Figure 2.2, users can use mobile, PDA, notebook and desktop to get the services from Oracle Application Server (OAS), and there are four main services, including communication services, business logic services, presentation services and caching services [17].

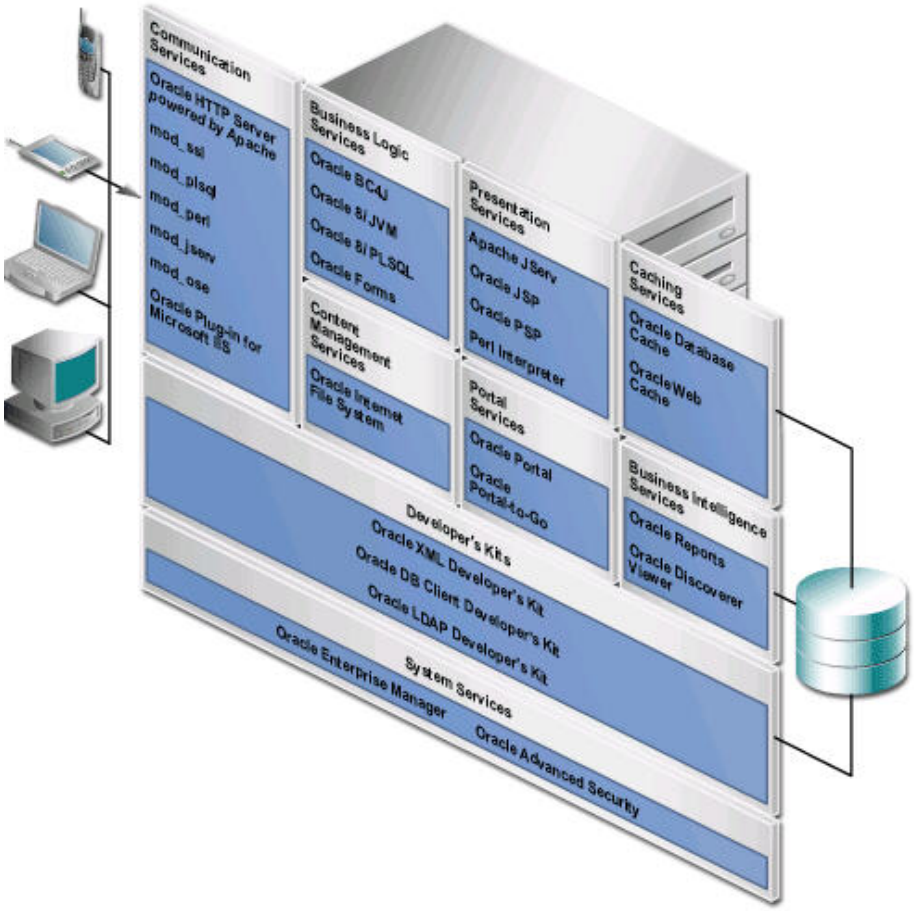


Figure 2.2. Oracle Application Server [17]

Communication services are responsible for the web resources dispatching between end-users, Business logic services are responsible for the interpreting the business logic to web server, Presentation services are responsible for the interpreting JSP, PSP, etc. languages to web server, and Caching service are responsible for the various caching mechanism including database caching, web cache for performance issue. In Figure 2.2, we can observe that it is difficult to diagnose the complicated module inside the OAS. In Figure 2.3, Oracle Database Server (ODS) is divided into three main components, including Memory, File and Daemon [18]. Memory is responsible for memory control for performance issue. File is responsible for information records. Daemon is responsible for database operation control. In Figure 2.3, we can observe that it is difficult to diagnose the complicated relationship between modules inside the database.

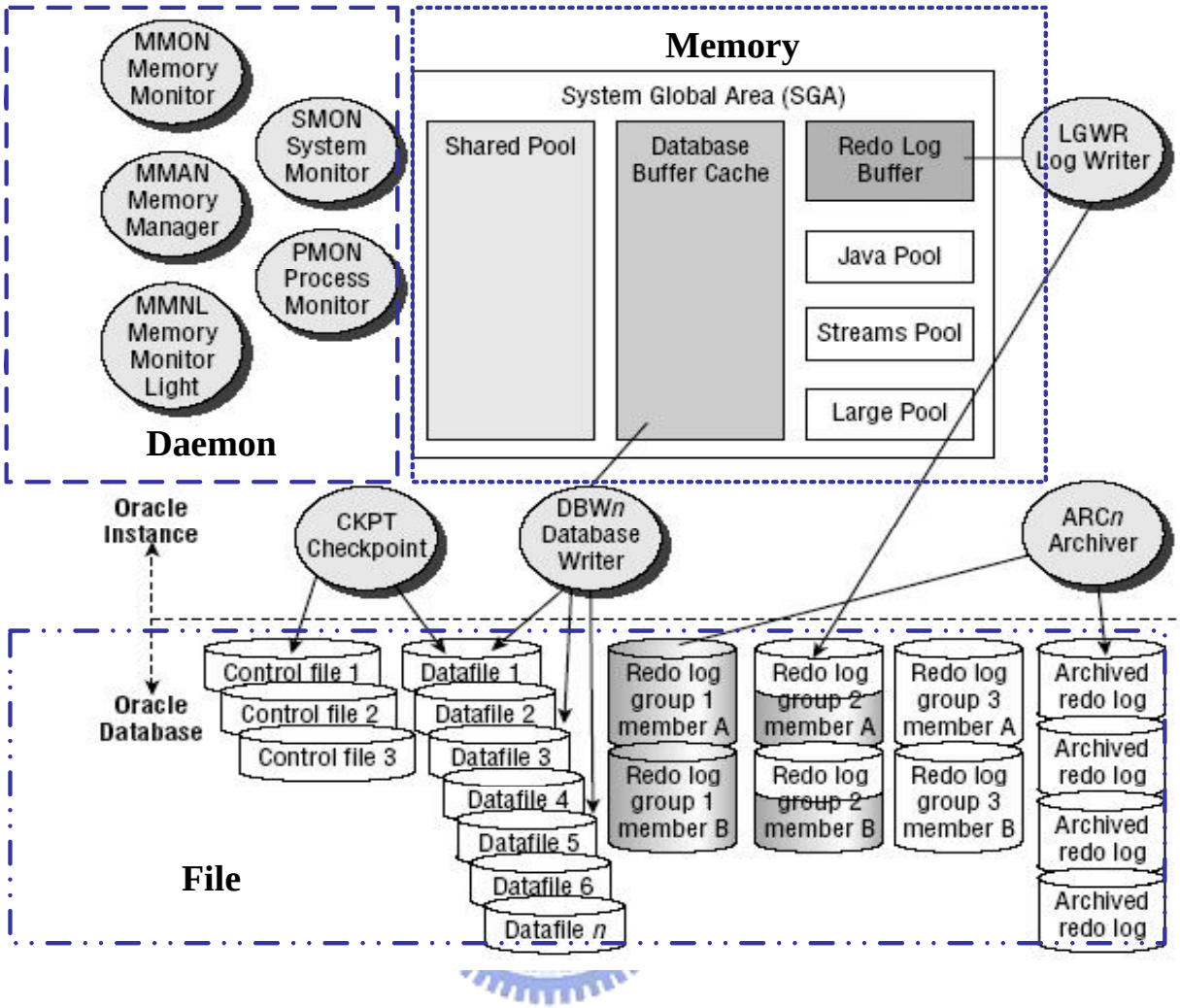


Figure 2.3. Oracle Database Server [18]

Traditionally, experts infer the system to find out the problems according to system characteristics; therefore we imitate experts to infer the problems by rule-based inference and use EMCUD [22] to elicit the embedded meaning while the system information can not be easily obtained.

2.2. Solution Retrieving

As for solution retrieving aspect, some other related researches provide an accurate attributes comparison by CBR [5] [10], but the system performance is low, due to lacking a

proper decision making mechanism. Those approaches only focus on similarity calculation precisely, and propose auto knowledge acquisition. Some papers propose the conversational case-based reasoning (CCBR) approach [4] to reduce comparison effect in attributes for cases retrieving; however once facing complex system in time limited domain, it may still be unable to bring functions into full play. Some papers propose the new technology in order to describe the scenario in multi-CBR architecture to handle complicated relationship between system modules [9], but still may fail to solve problem efficiently.

As we know, once employees who lack enough experience to solve problems, they usually face a lot of troubles and need to retrieve the corresponding solution; hence the approach case-based reasoning, which uses case retrieve, case reuse, case revise and case retain approaches to help employees to find the suitable solutions is proposed. However, usually each problem has at least one solution, so it is difficult to find all solutions.

Traditionally, when experts want to find the desired solutions for a given problem, they usually find out the pre-stored answer of similar problems. Assume the problems can be described by their attributes; using CBR as our approach to retrieve cases, reuse similar cases, retain the related information into case base, and revise attributes or attribute values in case base seems to be a good approach.

Chapter 3. The Design of Solution-Retrieval System

As we know, many traditional approaches, e.g., search engine or Knowledge Management (KM) center may be used for trouble shooting. However, those approaches may fail due to resulting too much information to help users to solve problems efficiently. In a complicated and time limited situation, such as critical system recovery, the accuracy is important.

In order to ensure the problems can be handled in limited time, we propose Solution-Retrieval System (SRS) to help on-duty employees.

3.1. Knowledge Representation

Trouble shooting is difficult for novices in some specific domains, e.g., Supply Chain related domains, which used Application server and DB as applications to support such a system. There are tree structures in modules of Application Server and DB. Therefore, based on system characteristics, we design the SRS by ontology approach.

As we know, the Oracle DB is composed of modules, modules have predecessor, successor and sibling relationships, e.g., the root module “Oracle DB” has two successors, “instance” and “database”, their common predecessor are “Oracle DB”, “instance” and “database” are sibling relationship, details are shown in Appendix C. The Oracle database modules are shown in Figure 3.1.

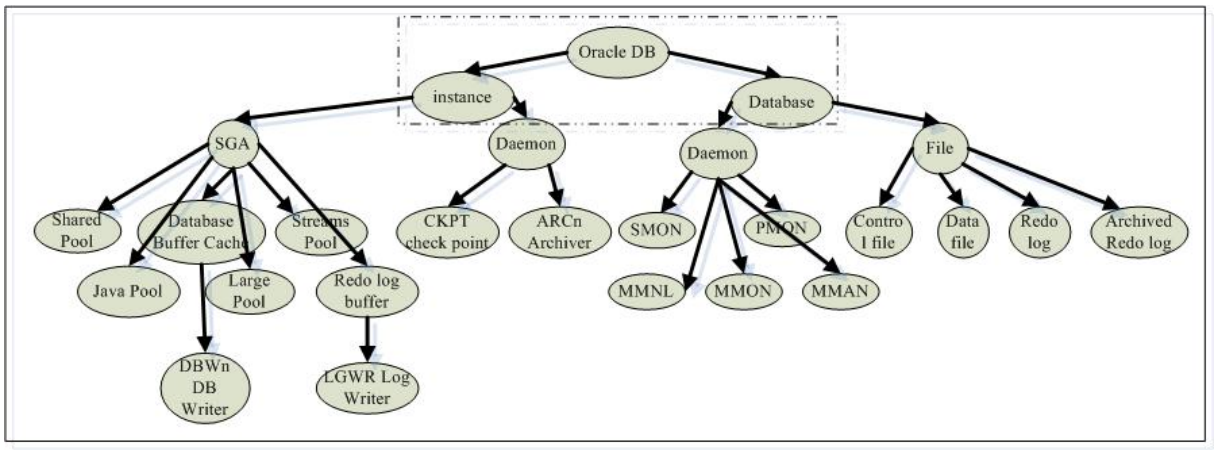


Figure 3.1. Database Module Tree

Since database modules have tree structure in nature, and the error type diagnosis is based upon the hierarchical structure to narrow down the error type. Therefore, as system is abnormal, experts can diagnose problems based on system characteristics to infer the error type.

Although knowledge acquisition (KA) is a bottleneck in expert system, Oracle consultants and DBA in our company make KA to be easily done. Besides, we use EMCUD [22] to elicit embedded meaning of the original knowledge when system information can not be easily obtained.

When problem occurs, experts usually use the following two steps to solve problems. Firstly, experts refer the error type from system. Secondly, they compare similarity between the problem and solution cases.

As we know, experts use their domain knowledge to narrow down the error scope, then compare the similarity between the narrowed down error scope (called the error type) and solved solutions according to some attributes, finally select high similarity cases as solutions to solve problems. Hence, we imitate experts to adapt hybrid knowledge base approaches to solve problems, as shown in Figure 3.2. In Phase one, experts infer the error type based on system characteristics, system status and related logs, e.g., inference from module “Oracle DB”, “Database”, “Daemon” then “SMON”, along with the root module “Oracle DB” to leaf

module “SMON” to diagnose the error type. Furthermore, experts reason the high similarity solutions to solve problems depending on the experience. The knowledge representation in trouble shooting is shown in Figure 3.2.

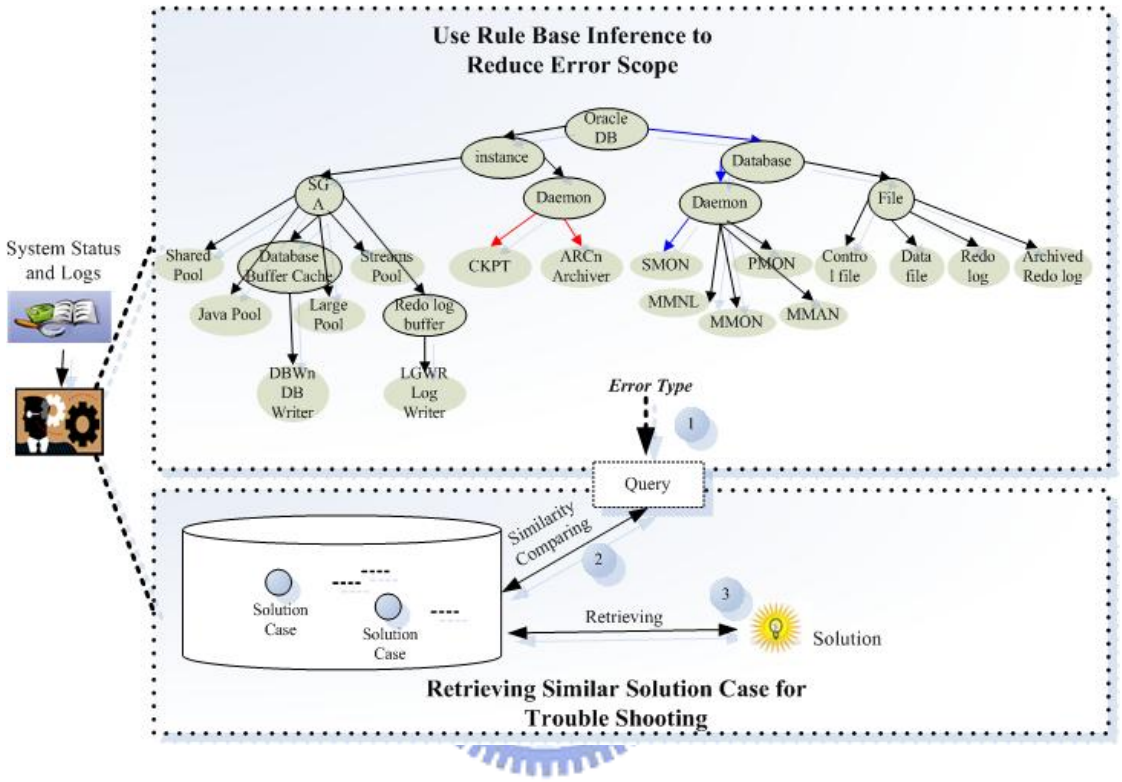


Figure 3.2. Knowledge Representation

In Phase one, we use **“If <condition> Then <action>”** condition statement as inference approach. Due to the varied system problems, we use NORM (New Object-oriented Rule Model), a forwarding chaining rule-based inference. The knowledge base (KB) can be divided into different knowledge classes depending on system characteristics, and the relations between classes include **Trigger, Acquire, Reference, and Extension-of**. Figure 3.3 displays the rule based classes for diagnosing Oracle DB.

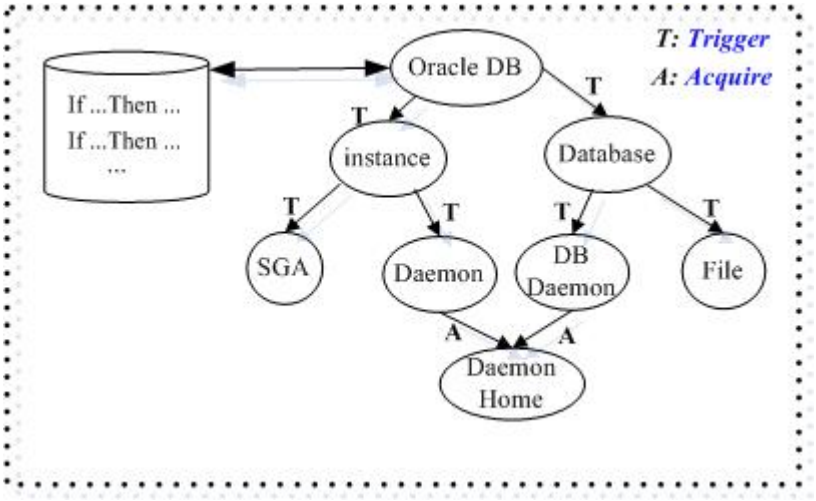


Figure 3.3. Rule base knowledge representation on Oracle database

In our approach, solution cases could be saved as data row consisting of several attributes in DB. In Phase two, we compare similarity between problems and cases using case-based reasoning for solution retrieving, e.g., problem subject, product version, product installed platform, error type, are for similarity calculation, and document type and revised date are for solutions ranking .

3.2. The SRS System Architecture

There are two phases in SRS. Phase one is the **problem diagnosis by rule based inference** with system logs, error logs and system status information as the input, and the error type as the output. Phase two **solution retrieving by case based reasoning** with query (the error type and user input) as the input and the related solutions as the output. The SRS is composed of three main components: preprocessor, rule-based inference module [22] and CBR module. Preprocessor is responsible for translating logs into facts, rule-based Inference module is responsible for inferring the error type, and CBR module is responsible for reasoning the related solutions. In Figure 3.4, totally there are ten steps to describe overall system operating flow by following scenario.

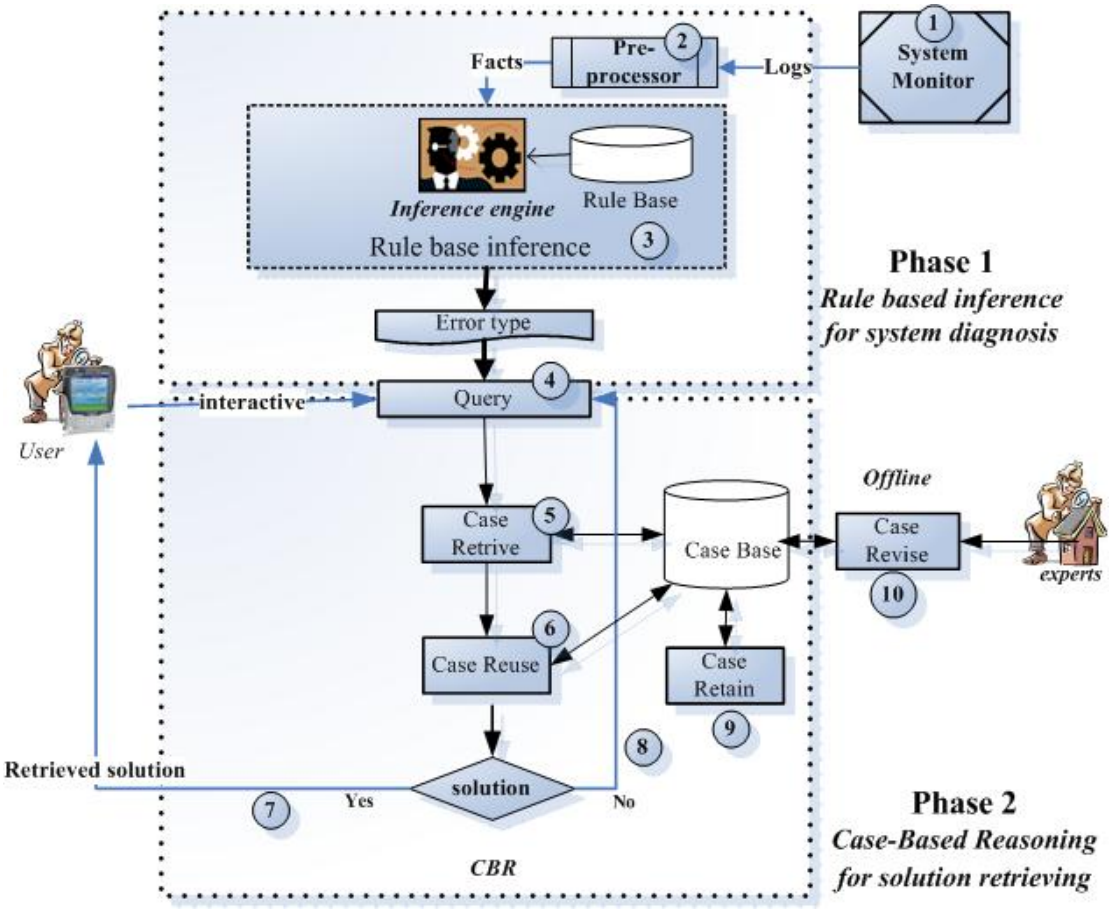


Figure 3.4. SRS Architecture Overview

Step 1. System Monitor: Once the system is abnormal, the system monitoring tool (called System Monitor) will detect the abnormal conditions and deliver related logs and system status to preprocessor for further process. The step is in on-line process.

Step 2. Preprocessor: The preprocessor will parse related logs and translate system status information into weight and facts, which can be used in adaptive queries and can be inferred by the inference engine, respectively. The step is in on-line process.

Step 3. Rule-based Inference: Via logs preprocessing, the inference engine will infer the facts and output the error type. The step is in on-line process.

Step 4. Query: Query is generated according to the obtained error type and users input. The step is in on-line process.

Step 5. Case Retrieve: CBR module will retrieve similar solutions from case base. The step is in on-line process.

Step 6. Case Reuse: The existing solutions can be reused when cases similarity is higher than a threshold value. The step is in on-line process.

Step 7. Retrieved Solutions: Feedback solutions to users while the solutions are found. The step is in on-line process.

Step 8. No Solution be Found: If there are no proper solutions could be found, users can try another keywords again. The step is in off-line process.

Step 9. Case Revise: Once no solutions could be found confirmedly, on-duty employees need to call experts for help, and the attributes or attribute values in case base need to be revised to fulfill the new problem requirements. The step is in off-line process.

Step 10. Case Retain: Retain revised value or information into case base to fit new requirements. The step is in on-line process.

In our design, **preprocessor** is responsible for transferring related logs and system status into facts. According to the different logs and system status, the preprocessor parses logs into different facts depending upon index table.

Example 3.1**. The Error Type of System Pending.**

1. As system is in a pending situation, the **preprocessor** retrieves system logs, e.g., “ORA-01575: timeout waiting for space management resource enquene”.

2. Based on keywords index table, preprocessor parses the logs contains keywords (e.g., “ORA”, “space management”, and “enquene”)

3. The facts are generated depending on parsed keywords, e.g., keyword **ORA** becomes fact “system=DB”, keyword **space management** becomes fact “module stuff=physical”, and “One Stuff” and keyword **enqueue** reveal the weight of “database version” is high.

Not only system logs but also system status which includes CPU, memory, and disk utilities, etc, are critical for problem reference to assist the diagnosis precisely.

Chapter 4. Problem Diagnosis by Rule-Based Inference

As we know, rules could be generated according to system modules, system characteristics, system logs ,and system status by knowledge acquiring from experts. We use NORM as knowledge representation and adapt forward chaining as our approach based on system characteristics.

4.1. Knowledge Acquisition

Based upon NORM concept, the rule base structure for system problem diagnosis is proposed. In Figure 4.1, the root rule class **System** consists of two rule classes **Application server** and **Oracle DB**, where **A** means acquire relation, **T** means trigger relation in rule classes.

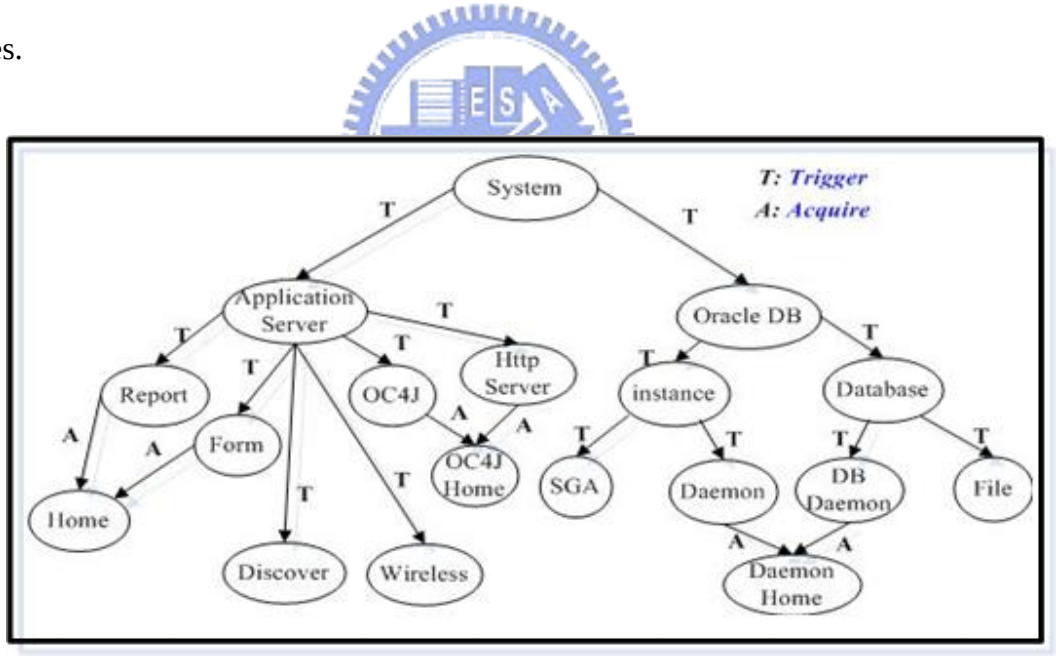


Figure 4.1. Rule Base Structure for System Problem Diagnosis

Definition 4.1**. Relation Acquire in Rule Classes.**

Sub-problem may be solved by acquiring another rule class, Once the acquired knowledge object ends inference process, it backs to original inference process.

Example 4.1**. Acquire System Status.**

Rule class **DB Daemon** acquires rule class **Daemon Home**, then backs to rule class **DB Daemon** and carries on inference job.

Definition 4.2**. Relation Trigger in Rule Classes.**

Some rule class is triggered when some specific conditions are satisfied. It means that a problem may be transformed into another problem.

Example 4.2**. Trigger Sub-Module.**

Rule class **Database** triggers rule class **DB Daemon** if the condition is satisfied.

Definition 4.3**. Acquisition Table (AT).**

AT is a repertory grid of multiple data types.

Boolean ：true or false

Single value：an integer, a real, or a symbol

Set of values：a set of integers, real numbers or symbols.

Range of values：a set of integers or real numbers.

The table of approach EMCUD [22], columns composes of objects, rows composes of facts. The corresponding object and fact have a value to identify the value of the object feature.

Example 4.3**. AT of Rule Class “Oracle DB”.**

Due to rule class “Oracle DB” containing objects “DB” and “Instance”, those objects are put in column title of AT. Facts are put in row title of AT to separate different objects, as shown in Figure 4.2.

Definition 4.4. **Attribute Ordering Table (AOT).**

Attribute ordering table (AOT), the table of approach EMCUD [22], columns

composes of objects, rows composes of facts. The corresponding object and fact have a value to identify the relationship between the object and the fact. **D:** means dominate the relationship, **X:** means no relationship, **integer** means the strength of relationship (from 1 to 5, 5 is the strongest relationship). According to database ontology, the inferring sequence is from top (the root-module) to bottom (the leaf-module).

Rule generation is composed of three steps described as follows. **Step** (1). Ontology: The system owns trigger and acquire relationships between rule classes depending on rule class features. **Step** (2) AT and AOT of EMCUD [22]: The objects are put in columns, and facts are put in rows to build objects values and relationships. **Step** (3) Rule generation: The rules are generated by AT and AOT, as shown in Figure 4.2.

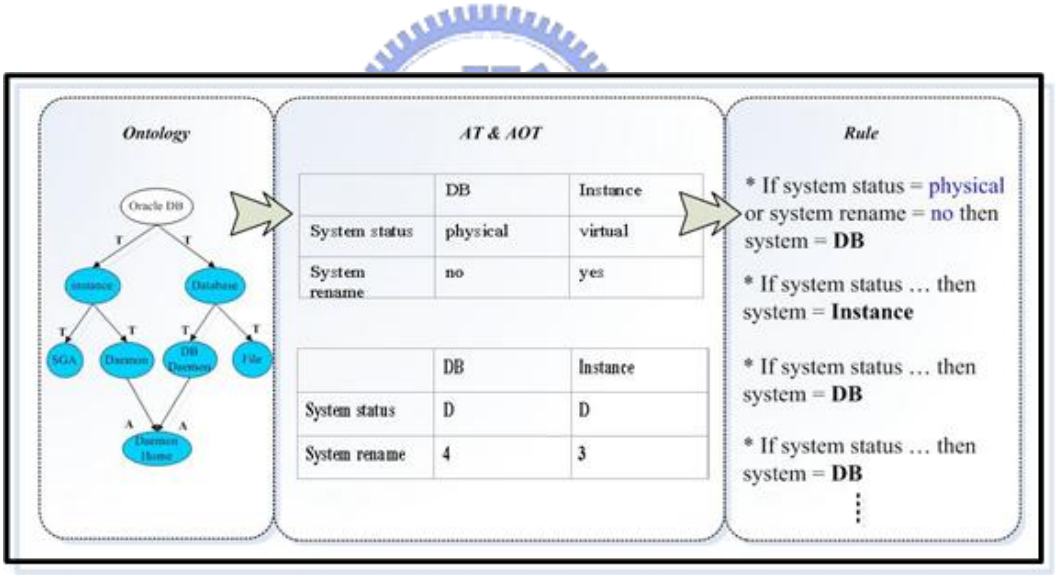


Figure 4.2. Rule Generation from Rule Class Oracle DB

Example 4.4**. Rule Generation by EMCUD.**

In Figure 4.2. Rules are Generated from Rule Class **Oracle DB**

(1) Rule class **Oracle DB** is the root class of the ontology.

(2) Based on the features of **Oracle DB**, objects in column contain **DB** and **Instance**, facts in rows include **System status** and **System rename**. The input values **physical**, **virtual,**

no, and **yes** in AT are listed from top to bottom, left to right in sequence to describe the feature values between each object and fact. Input values are **D, D, 4, 3** in AOT to describe the relationships strength between each object and fact. The objects **DB** and **Instance** are described as follows:

- DB: The module of database which is persistent.
- Instance: The module of database which is virtual.

The facts are listed as follows:

- System status: The system status is the fact which describes system status.
- System rename: The system rename is the fact which means whether the system could be renamed or not.

(3) The rules are generated from AT and AOT, as shown in Figure 4.3.

```
IF (System stauts=physical) AND (System rename=no) THEN DB CF=0.8
IF (System stauts=physical) AND NOT(System rename=no) THEN DB CF=0.4
IF (System stauts=virtual) AND (System rename=yes) THEN Instance CF=0.4
IF (System stauts=virtual) AND NOT(System rename=yes) THEN Instance CF=0.6
```

Figure 4.3. Rules example in rule class Oracle DB

After inferring root class **Oracle DB**, the inference engine will determine next rule class. Based on the features of the next rule class **Database** which was inferred by **Oracle DB**, both the AT and AOT are produced and rules are generated by those tables, example is shown in Figure 4.4. The object **daemon** and **file** are described as below:

- Daemon: The daemon is the DB module which keeps the database alive and do periodically system check.
- File: The file is the system files which contains system files to control and record system status.

The facts are listed as below:

- Module stuff: The module stuff is the stuff that describes modules status.

- Module behavior: The module behavior is the behavior which describes modules working behavior.
- Module number: The module number is the number which describes the module number.

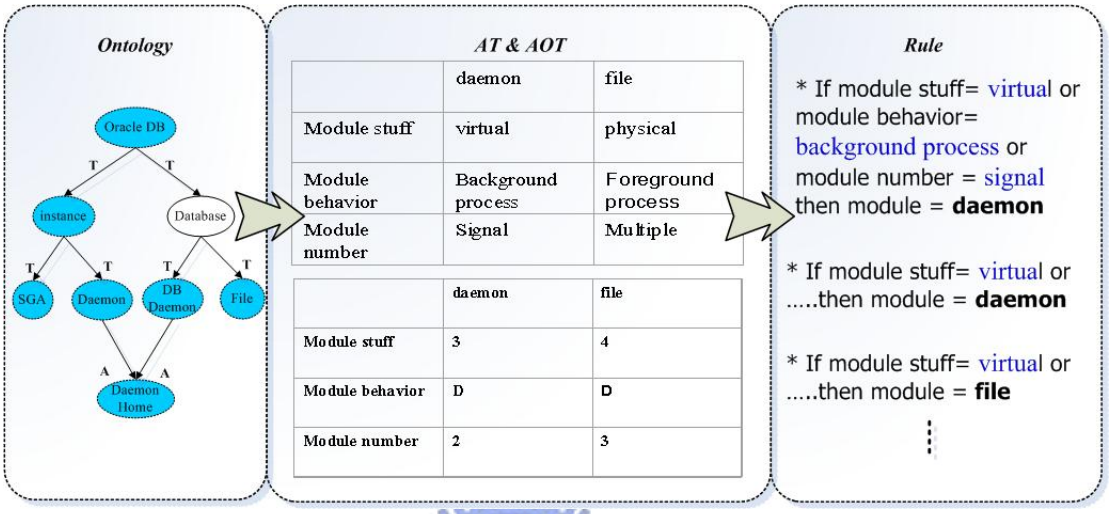


Figure 4.4. Rule Generation from Rule Class Database

Finally the rule class **DB daemon** is triggered, and the rule class **Daemon home** will be acquired on next step, and go back to rule class **DB daemon**. The rule class **Daemon home** describes system information. Appendix A shows AT and AOT of rule class **Daemon home**. The rule class **DB daemon** describes database daemon. Appendix B shows the details of AT, AOT of the rule class **DB daemon**.

4.2. Embedded Rules

Since system information are not always obtained; using EMCUD [22] to elicit embed meaning seems useful and the error type could be inferred.

Definition 4.5. **EMCUD.**

Embedded Meaning Capturing and Uncertainty Deciding (EMCUD). A New Model for Eliciting. Knowledge Representation combine AT (or Conventional Repertory grid) and

AOT.

Example 4.5**. EMCUD.**

Figure 4.5 shows the EMCUD inference processes, which are composed of two parts (Error type diagnosis and Embedded rule generation). The error type diagnosis is composed of AT and AOT in rule class “Daemon Home” as shown in Figure 4.4. In Figure 4.5, there are five different inference conditions.

(A). Transaction abort.

In Oracle database, describe the database transaction situation. Transaction abort is caused by the disconnected connections between client side and server side.

(B). Not commit transaction.

In Oracle database, when a unconfirmed transaction has been modified (insert, update or delete).

(C). Transaction Number.

In Oracle database, transaction number is the count of online processes. It will cause an abnormal situation in DB when transaction number is overloading.

(D). CPU loading.

Central Processing Unit (CPU) is the hardware device in server for calculation and control other related units. When CPU utility is overloading, It will cause an abnormal condition.

(E). Memory Loading.

Memory is the hardware device for data high speed calculation and caching buffer. Usually, memory loading is the important reference fact during problem diagnosis.

Object Definition. TROLL. Transaction Rollback.

In Oracle database, TROLL means unconfirmed transactions need to be rolled back

due to related transaction abort or error.

In normal condition, inference engine refers the error type, which contains higher certainty factor (CF) by using EMCUD. e.g., rule **IF** A & B & C & D & E **Then** T with high CF, T is the inference result. As system information can not be easily obtained in problem diagnosis, we use EMCUD [22] to elicit the embedded meaning. e.g., rule **IF** A & B **Then** T with lower CF, T is the inference result.

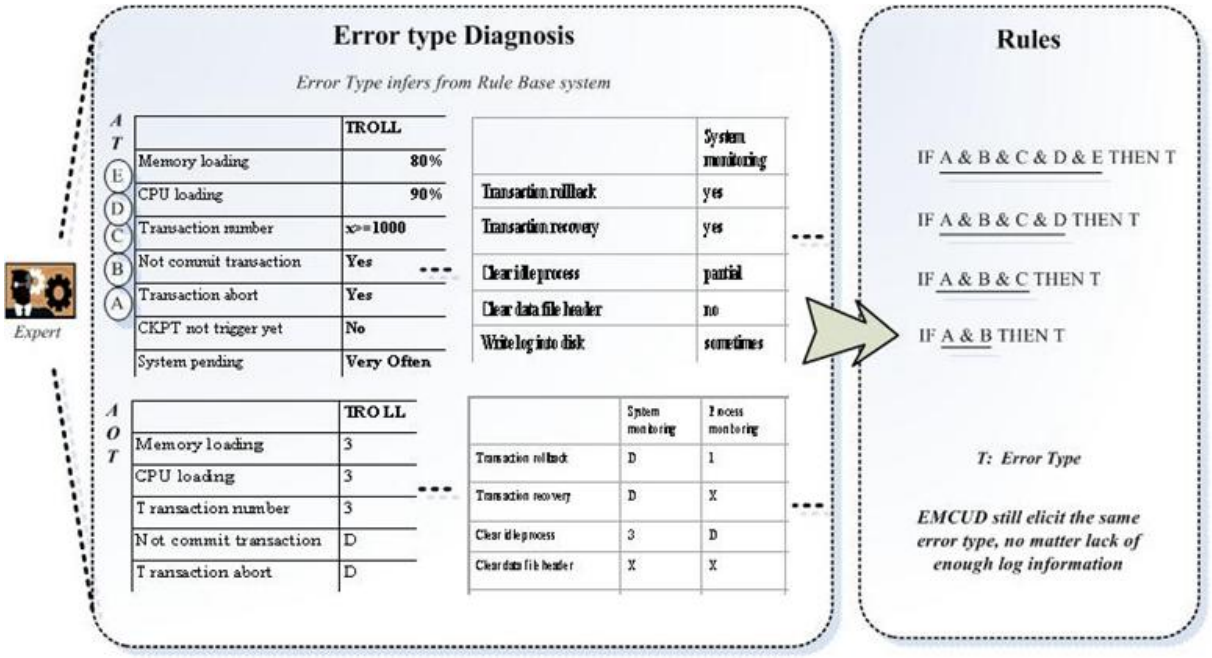


Figure 4.5. EMCUD

Chapter 5. Solution Retrieving by CBR

After problem diagnosis in phase one, the query is generated according to the obtained error type and users input. In Phase two, our goal is using CBR approach to find similar cases, and use the retrieved solutions to solve problems.

The output in phase consists of the error type and weights, the error type could be used to increase performance in similarity calculation, and weights could be used to adjust weights in case attributes to increase precision, as shown in Figure 5.1.

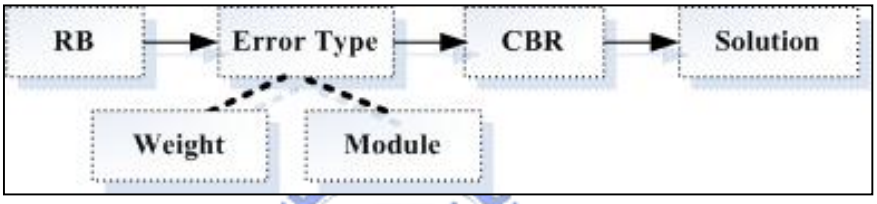


Figure 5.1. Adaptive query in CBR

5.1. Case Retain

As we know, the error type contains multi-attributes, e.g., problems subjects, module name (application), installed platform (platform) and product version (version). Therefore, 12 attributes are chosen as the representative of CASE_ID, NAME, SUBJECT, VERSION, PLATFORM, APP, DOC_TYPE, REVISION_DATE, PRODUCT, WORD_VECTOR, SOLUTIONS, SOLUTION1, as shown in Figure 5.2.

Name	Type
CASE_ID	NUMBER
NAME	VARCHAR2(30)
SUBJECT	VARCHAR2(400)
VERSION	VARCHAR2(30)
PLATFORM	VARCHAR2(50)
APP	VARCHAR2(50)
DOC_TYPE	VARCHAR2(50)
REVISION_DATE	DATE
PRODUCT	VARCHAR2(30)
WORD_VECTOR	VARCHAR2(30)
SOLUTIONS	VARCHAR2(2048)
SOLUTIONS1	LONG

Case attributes (*subject, version, platform, app, Doc_type, Revision_date*)

Figure 5.2. Case Attributes representation in case base

Furthermore we use two attributes : “document type” and “revision date” to rank the obtained solutions. Figure 5.3 shows an example of case in case base.

SUBJECT	APP	VERSION	PLATFORM	DOC_TYPE	REVISION	SOLUTIONS
ORA-1575 Timeout waiting	smon	8.1.7	Solaris			1. Re-start the ins
ORA-01575 ORA-01662 Ti	smon	9.0.2	Solaris	How to	2005/3/15	If the database is
ORA-1575: Timeout waitin	smon	7.2.3	Unix Dec Al			These errors sho
SMON - Temporary Segm	smon			FAQ		The ORA-1575 is
ORA-01575: timeout waitin	smon	9.2.0.4.0	SunOS 5.6	Forum		Explanation: This
ORA-9999 error and SMOI						Note, TEMPORA
ORA-1575 Timeout waiting					2004/5/18	
SMON: Restarting fast_sta	smon	9.2.0.6	Solaris 8	Forum		
Dropping Tablespace Ty	smon			PROBLEM	2003/5/23	
ORA-9999 error and SMOI	Smon & pmon	10.1.2	NT	Forum	2005/10/1	

Figure 5.3. Case example

In Figure 5.3, red rectangle dotted line contains 4 attributes (subject, app, version, platform) which contained in both are query and case, where each case has a corresponding solution (blue rectangle dotted line) on different attributes (e.g., subject, app, version and platform could be integrated as an unique combination to identify different solutions.). Since those attributes are comparable between query and cases, we use equations to calculate their similarity the weights of the attributes can be adjusted due to different logs and system information in Oracle database and Oracle Application Server.

5.2. Case Retrieve

Definition 5.1. **Similarity Equation between Query and Case.**

$$\begin{cases} S(q,c) = W_s \times S_s(q,c) + W_a \times S_a(q,c) + W_p \times S_p(q,c) + W_v \times S_v(q,c) \\ W_s + W_a + W_p + W_v = 1 \end{cases}$$

(1) The attributes Subject, Application, Platform and Version are used to identify the similarity between query and case in case base, different logs and system status will form different weights in each attribute designed by experts. The calculation method is shown in Definition 5.1. **Similarity Equation between Query and Case**.

(2) In the equation, where

$S(q,c)$	the similarity between query and case
$S_s(q,c)$	the similarity between query and case in subject attribute
$S_a(q,c)$	the similarity between query and case in application attribute
$S_p(q,c)$	the similarity between query and case in platform attribute
$S_v(q,c)$	the similarity between query and case in version attribute
W_s	subject weight
W_a	application weight
W_p	platform weight
W_v	version weight

(3) As we know, both error type and case have 4 attributes to compute similarity.

Definition 5.2**. Similarity Equation of Subject.**

$$S_s(q,c) = \begin{cases} \frac{N_{\min}}{N_{\max}} \times \frac{W_m}{N_i} \times W_i, N_{\max} = \{N_i, N_c\}, N_{\min} = \{N_i, N_c\} & , i = 1 \\ \sum_{i=1}^{n} \frac{N_{\min}}{N_{\max}} \times \frac{W_m}{N_i} \times \frac{(N_i - W_i)}{N_i - 1}, W_m = \{0,1\} & , i > 1 \end{cases}$$

where

$N_{\min}$	Min number of Keywords of query and case
$N_{\max}$	Max number of Keywords of query and case
W_m	Keyword mapping weight 0 or 1
N_i	the number query keywords
W_i	Keyword weight
N_c	The number of case keywords

(1) Subject is the short description to describe the error situation. Firstly, the attribute "subject" could be translated as keywords based on keywords table. The similarity is different based upon different keywords and the keywords ranking order in sequence, as shown Figure 5.4.

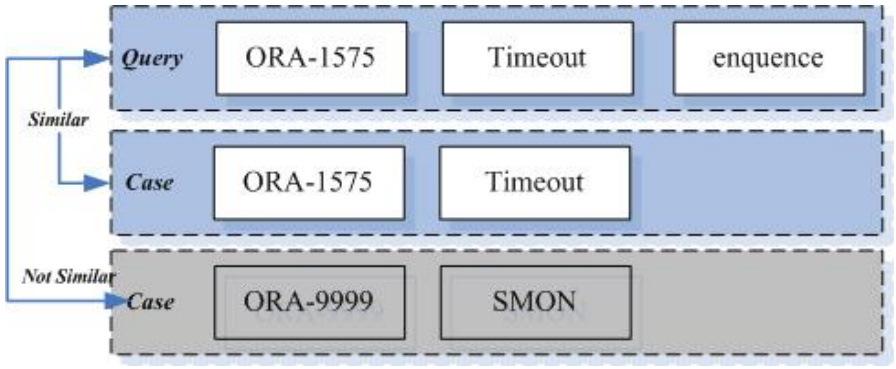


Figure 5.4. Example of query and case comparison

(2) In Figure 5.4, query and first case are similar, since they have the same keywords (e.g., ORA-1575 and Timeout) and the same ranking order. From this idea, the calculation equation was derived, as shown in Definition 5.2. **Similarity Equation of Subject**, where $S_s(q,c)$ is the similarity between query and case in subject attribute.

Example 5.1**. Subject Similarity.**

Similarity example was shown as below.

Similarity : $S_s(q,c) = \frac{2}{3} \times \frac{1}{2} \times \frac{3}{2} + \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2} = 0.66$ Query and Case pairs in Blue

$S_s(q,c) = \frac{2}{3} \times \frac{0}{2} \times \frac{3}{2} + \frac{2}{3} \times \frac{0}{2} \times \frac{1}{2} = 0$ Query and Case pairs in Gray

In Example 5.1, query and first case are similar, where the similarity is 0.66 (the highest value is 1 in similarity value).

Definition 5.3**. Similarity Equation of Application.**

$$\begin{cases} S_{ad}(q,c) = D_{distance}(q,c) + C_{constance}(q,c) \times D_{dicrection}(q,c) \\ S_a(q,c) = \dfrac{D_{\max} - S_{ad}(q,c)}{D_{\max}} \end{cases}$$

where

$$\begin{cases} S_{ad}(q,c) & \textit{Similarity distance} \\ D_{distance}(q,c) & \textit{Distance between query and case} \\ C_{constance}(q,c) & \textit{Constance between query and case} \\ D_{dicrection}(q,c) & \textit{Direction change times} \\ D_{\max} & \textit{Max distance} \end{cases}$$

(1) Application is the inferred module name in problem diagnosis phase, which is the root cause of the error occurred, e.g., SMON (system monitoring). Since database modules are composed of tree structure, the attribute “application” could be translated as ontology model, called application ontology, as shown in Figure 5.5.

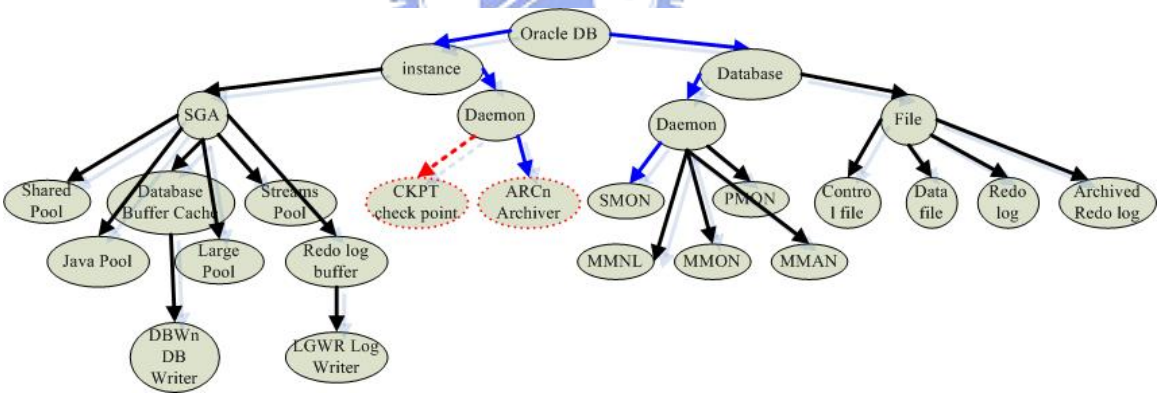


Figure 5.5. Application ontology

(2) In Figure 5.5, using application ontology to determine similar pairs from modules in database, e.g., the distance of modules CKPT and ARCN is closer than modules ARCN and SMON; hence the similarity of modules CKPT and ARCN is higher that modules ARCN and SMON.

(3) In the equation, where $S_a(q,c)$ is the similarity between query and case in app attribute.

Example 5.2**. Application Similarity.**

Application similarity example is shown as below.

Similarity distance:

$$S_{ad}(q,c) = 6 + 0.5 \times 1 = 6.5$$
$$S_{ad}(q,c) = 2 + 0.5 \times 1 = 2.5$$

Similarity:

$$S_a(q,c) = \frac{7.5 - 6.5}{7.5} = 0.13$$
$$S_a(q,c) = \frac{7.5 - 2.5}{7.5} = 0.66$$

In Example 5.2, firstly, calculate distance in modules, secondly, using similarity equation to determine similarity. The similarity between two modules (SMON and ARCN) is 0.66, which is higher than the similarity value 0.13 between modules CKPT and ARCN.

Definition 5.4**. Similarity Function of Platform.**

$$\begin{cases} S_{pd}(q,c) = \alpha \,|\, C_{platform} - Q_{platform} \,|\, + \beta \,|\, C_w - Q_w \,| \\ S_p(q,c) = \dfrac{|\, D - S_{pd}(q,c) \,|}{D} \end{cases}$$

where

$S_{pd}(q,c)$	Platform distance	$C_{platform}$	Platform weight of case
$S_p(q,c)$	Platform similarity	$Q_{platform}$	Platform weight of query
D	Constant	C_w	Version weight of case
		Q_w	Version weight of query
		α	Constant
		β	Constant

(1) The platform definition is the installed platform of the Oracle DB and Application

Server. The platform is the attribute to compute similarity in solution cases. For this reason, the platform in our example are divided into Windows NT, Red hat Linux, Sun Solaris and IBM AIX, the sub revision in versions could be separated in each different platform. According to this concept, the platform representation could be composed of a tree structure, named Weight Platform Tree, as shown in Figure 5.6.

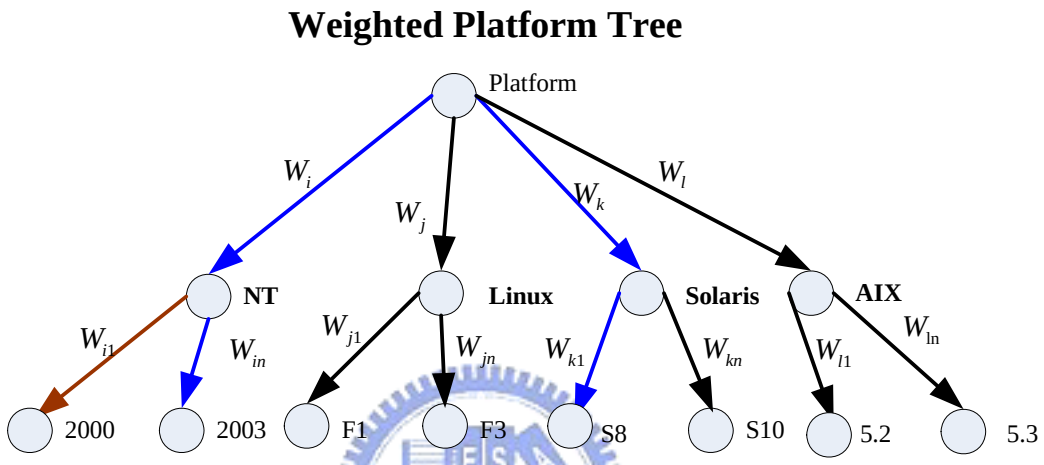


Figure 5.6. Weight Platform Tree

(2) In the equation, where W_i, W_j, W_k, W_l are platform weights, $W_{i1} \sim W_{in}$ are the NT platform sub revision weights, $W_{j1} \sim W_{jn}$ are the Linux platform small revision weight, $W_{k1} \sim W_{kn}$ are the Solaris platform small revision weight, and $W_{l1} \sim W_{ln}$ are the AIX platform small revision weight. The weights W_i, W_j, W_k, W_l are greater than other small version weights, which adjust by experts.

(3) The equation is shown in Definition 5.4**. Similarity function of platform**, where $S_p(q,c)$ is the platform similarity between query and case.

Example 5.3**. Platform Similarity.**

The example is shown as follows.

Similarity distance :

$$S_{pd}(q,c) = \alpha\,|0.5-0.5| + \beta\,|0.02-0.01| = 0.005$$
$$S_{pd}(q,c) = \alpha\,|0.7-0.5| + \beta\,|0.05-0.02| = 0.215$$

Similarity :

$$S_p(q,c) = \frac{|D-0.005|}{D} = 0.995$$
$$S_p(q,c) = \frac{|D-0.215|}{D} = 0.785$$

where $\begin{cases} \alpha = 1, \beta = 1/2 \\ D = 1 \end{cases}$ *, The value were adjusted by experts*

In Example 5.3, firstly, calculate similarity distance based on platform version tree structure, secondly, using similarity equation to determine similarity. The similarity of platforms, (e.g., NT 2003 and Solaris 8) value is 0.785 , which is lower than similarity value 0.995 in platform versions (e.g., NT 2000 and NT 2003).

Definition 5.5**. Similarity Function of Version.**

$$\begin{cases} S_{vd}(q,c) = \gamma\,|C_{version} - Q_{version}| + \lambda\,|C_{vw} - Q_{vw}| \\ S_v(q,c) = \dfrac{|D_v - S_{vd}(q,c)|}{D_v} \end{cases}$$

where

$S_{vd}(q,c)$	Version distance	$C_{version}$	Main version weight of case
$S_v(q,c)$	Version similarity	$Q_{version}$	Main version of query
D_v	Constant	C_{vw}	Sub version weight of case
		Q_{vw}	Sub version weight of query
		γ	Constant
		λ	Constant

(1) The **version** definition is the installed version of Oracle DB and Application Server. Except for subject, application, and platform, the product version is still an attribute. Oracle DB versions have tree structure stuff; as shown in Figure 5.7.

Weighted Database Version Tree

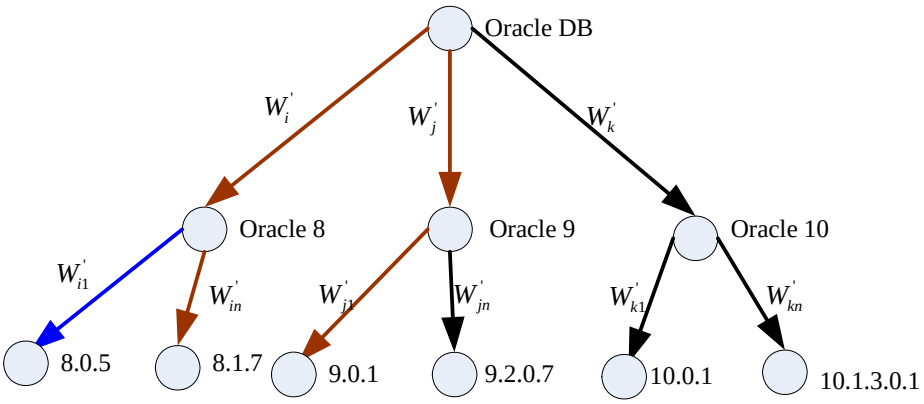


Figure 5.7. Weighted database version tree

(2) Nowadays, the most common used version are Oracle 8, 9 and Oracle 10g in Oracle database. Each version has its own sub version, e.g., Oracle 9 have 9.0.1 ~ 9.2.0.7 sub versions. The main version e.g., the weight of Oracle 8, W_i' is higher than the sub versions, e.g., 8.0.5 W_{i1}', weights are designed by experts.

Example 5.4. **Version Similarity.**

Version similarity was shown as below.

Similarity distance:

$$S_{vd}(q,c) = \gamma\,|0.9-0.8| + \lambda\,|0.03-0.02| = 0.11$$
$$S_{vd}(q,c) = \gamma\,|0.8-0.8| + \lambda\,|0.02-0.01| = 0.01$$

Similarity:

$$S_v(q,c) = \frac{1-0.11}{1} = 0.89$$
$$S_v(q,c) = \frac{1-0.01}{1} = 0.99$$

In Example 5.4, firstly, calculate similarity distance since database versions have tree structure in nature, secondly, using similarity equation to determine similarity. The similarity between two versions (Oracle database 9.0.1 and 8.1.7) is 0.89, which is lower than similarity value 0.99 in versions Oracle database 8.0.5 and 8.1.7.

As we know, different log will trigger different weight, e.g., log pattern “enquence” means “version weight” need to adjust higher value by experts. e.g., $S(q,c)$ = 0.3 * $S_s(q,c)$ + 0.25* $S_a(q,c)$ + 0.05 * $S_p(q,c)$ + 0.4 * $S_v(q,c)$, where 0.4 is the version weight.

Example 5.5. **Case Similarity.**

Case 1: $S(q,c)$ = 0.3x0.66+0.25x0.66+0.05x0.995+0.4x0.99 = **0.808**

Case 2: $S(q,c)$ = 0.3x0.00+0.25x0.13+0.05x0.785+0.4x0.89 = **0.427**

In Example 5.5, the similarity values are 0.808 and 0.427 in case1 and case2, respectively. From experiment, case 1 is similar than case 2 in the problem “system pending”.

Chapter 6. Implementation and Evaluations

In our experiment, the main operating system deployed is Microsoft 2003 and Solaris 8; the expert system tool is DRAMA enterprise 2.5 [22]; the application server is Oracle Application Server 10g (9.0.4) [19]; the database server is Oracle database (9.2.0.1) [20]; the implementation of SRS is in Oracle Jdeveloper 10g [21]. The Small device platform simulator is Windows CE 5.0 with 320 by 240 dimensions.

In the following, we will focus on the system problem diagnosis module (Rule-Based Inference) and solution retrieving module (Case-Based Reasoning). Figure 6.1 displays the user interface of SRS.

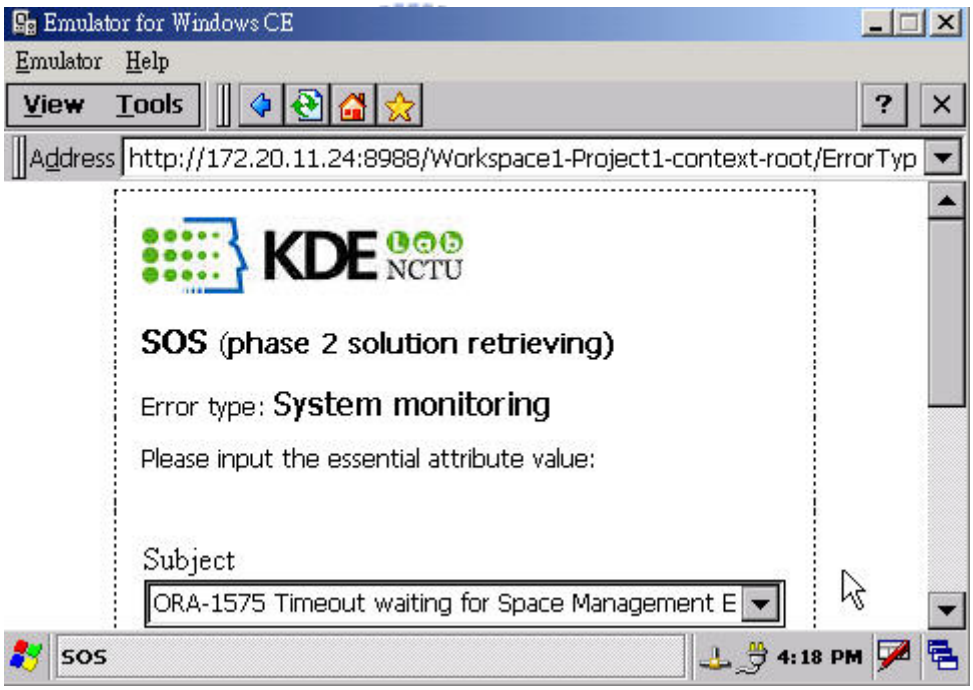


Figure 6.1. SRS User Interface

6.1. Implementation of SRS

When the on-duty employees receive the error warning from short message server, traditionally they use KM Center or Search Engine as their approaches to solve problems, e.g., “Metalink” (https://metalink.oracle.com) or Google (http://www.google.com). Although they found the solution eventually, the processing of system diagnosis and solution retrieving are still not efficient due to lack of domain knowledge and experience. Therefore, we propose the SRS to assist on-duty employees to deal with those problems efficiently. SRS is composed of two main modules, problem diagnosis module and solution retrieving module.

In problem diagnosis module, we use rule-based inference approach which imitates expert’s inference model [22]. In other words, SRS provides a system problem diagnosis module. As system is abnormal, monitoring daemons will collect error logs and system information and trigger preprocessor to translate related logs into the facts, inference module [22] continues to infer the possible error types.

In solution retrieving module, we use CBR approach to calculate similarity between query and cases in case base. In other worlds, SRS provides a solution retrieving module that assist users to find out the proper solutions.

When the error type is referred by problem diagnosis module, then the solution retrieving module will retrieve the cases, reuse the solution cases and feedback solution to users. When no proper solution is found, on-duty employees will call expert for help. The experts will be on site to solve problem, and revise the attribute values or attributes in case base to full fill the needs of the new problem.

Example 6.1. System Problem Diagnosis Example of Database Pending.

We use database pending as an example and capture the symptoms of system pending as inference information. In SRS, users will receive the diagnosed error type [1][15] by small

devices, then user can input keywords to search the solutions, as shown in Figure 6.1 and Figure 6.2.

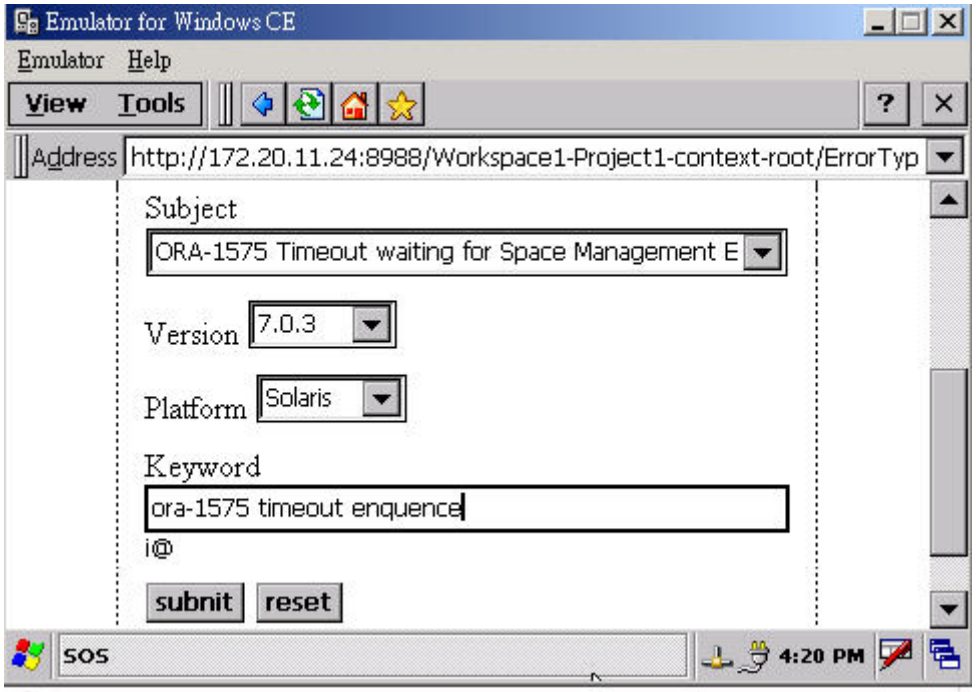


Figure 6.2. SRS User Interface in Query

After submitting the query, the solution lists will be displayed [13]. Each solution contains **subject**, **application, version and platform**, and the probabilistic similarity will be displayed by descending list, as shown in Figure 6.3. Therefore, users can choose solutions depending on probabilistic similarity [2] [11].

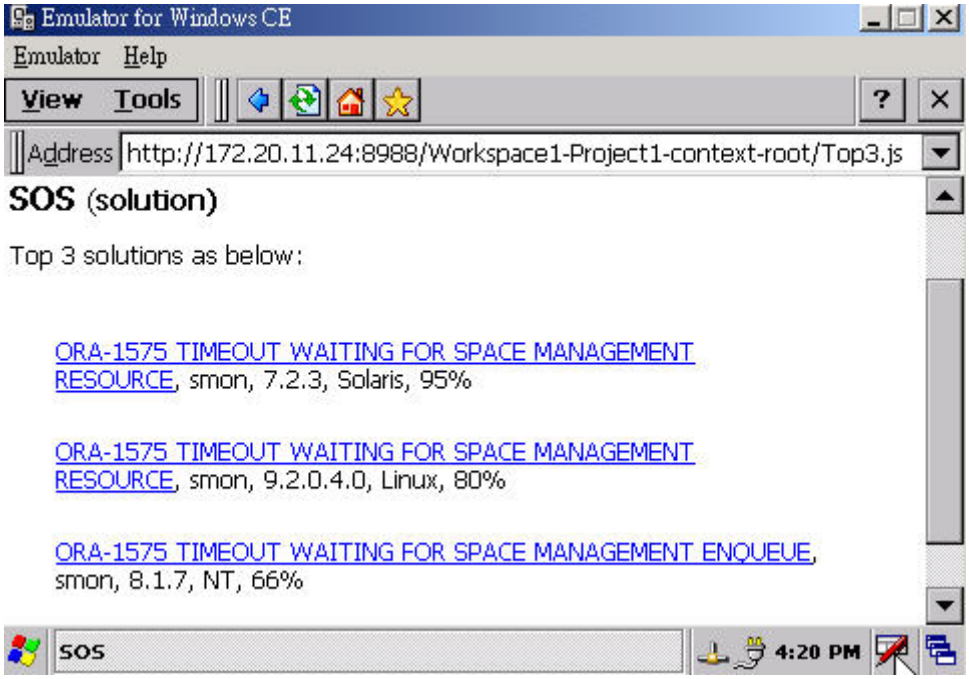


Figure 6.3. Solution List in SRS

Eventually, the solution will show the know-how and assist users step by step to solve system problems efficiently, as shown in Figure 6.4 [3].

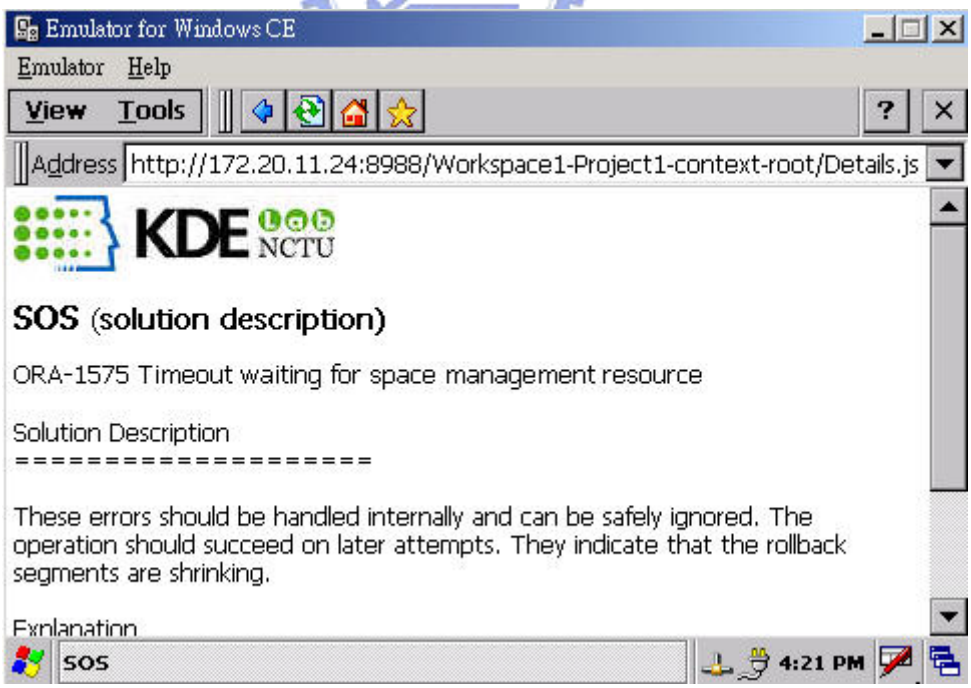


Figure 6.4. Solution Description in SRS System

6.2. Evaluation of SRS

In this thesis, we use questionnaire to evaluate our system. Firstly, we find the expert in our company to diagnose problems, and we find five users in our company to test the SRS. Finally, we use 6 kinds of predefined problems to test the accuracy and performance between SRS and KM center. Hence, both the users satisfaction and SRS accuracy could be evaluated. In this experiment, we use about 10800 rules to infer the error type, and use 36 real cases in case base that contains six kinds of error type and the corresponding solutions. We use 54 keywords in index table to assist preprocess precisely. SRS retrieves top 10 solutions list to assist employees trouble shooting; hence we compare probabilistic similarity toward 6 kinds of problem. Details are described in Table 6.1

Table 6.1

Error Types of System Diagnosis

Error Type	***Description***
db crash	Db crash means database crashed and unable to startup, this problem could cause other related issues, e.g., missing data integrity.
missing redo log	Redo log is a buffer that save data in memory for data caching, once missing redo log, this database will crash immediately.
archive log	Archive log is for data compression in database, once archive log is missing or crashing will cause data compression issue.
Data file	Data file is the file for saving data records those are committed in database. Once data file is missing or crashing, it will cause data lost and database crash.
control file	Control file is for controlling data file profile and related database parameters setting. Once control file is missing or crashing, it will cause database crash
system monitoring	System monitoring (SMON) performs instance recovery following an instance crash, combine free spaces in database, and managed spaces used for sorting. Once SMON is crashing, it will cause database crash immediately.

Experiment Case. Solutions list probability on similarity towards 6 kinds of error type

SRS training cases are listed in Table 6.2 which shows solutions list probability on similarity between solution cases towards 6 kinds of error type.

Table 6.2

SRS system toward 6 kinds of error type on similarity

Top3 Lists	Error Types					
	db crash	Missing redo log	archive log	data file	control file	system monitoring
top1	64.50%	87.00%	87%	70.00%	92.00%	93.00%
top2	59.00%	61.00%	15.00%	50.50%	35.00%	78.50%
top3	59.00%	15.00%	9.50%	32.00%	9.50%	58.00%

The distributed probability is shown in Table 6.2. From the experiment result, top 3 solutions list in error type “db crash” is closer on accuracy. In error type “control file” and “system monitoring” of top1 solutions probability are more accurate than others. On average, SRS system solutions probability is over 60 percent.

Evaluation 1. Problem solving on accuracy between SRS and expert

Since occurred keywords are changeable in real cases; hence we use Table 6.3 to describe experiment the number of keywords, test times, KM Center hit the problem times in solution top 10 list, SRS hit the problem times in solution top 10 list to compare hit ratio of problem solving accuracy between SRS and expert. Besides, test times are based on keywords to set different numbers of keyword.

Table 6.3

Problem Solving on Precision

Evaluation Entities	**Error Types**					
	db crash	missing redo log	archive log	data file	control file	system monitoring
Test times	5	4	6	4	4	5
KM Center hit	3	3	4	2	2	3
SOS hit	4	4	5	3	3	4
KM Center accuracy	60%	75%	66%	50%	50%	60%
SOS accuracy	80%	100%	83%	75%	75%	80%

The diagram of precision evaluation table (Table 6.3) is shown in Figure 6.5.

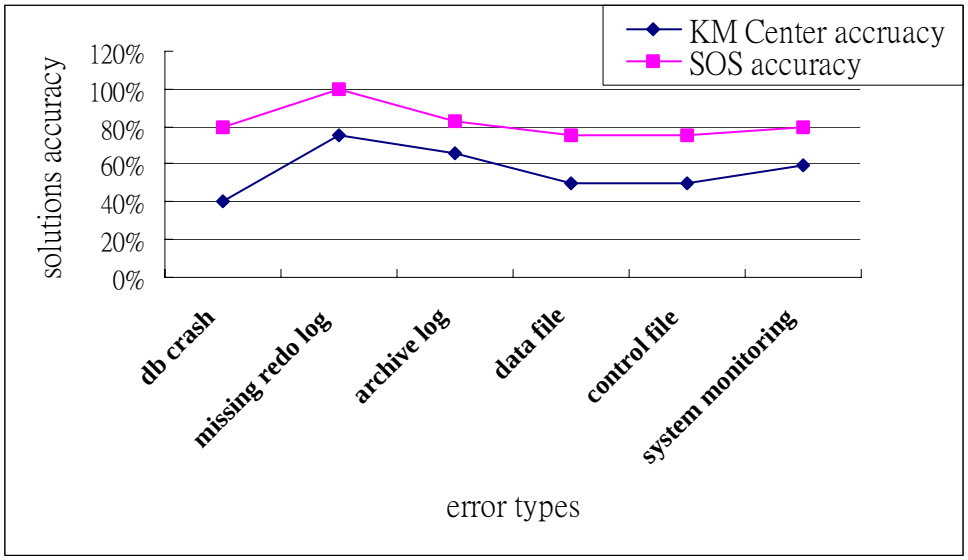


Figure 6.5. Problem Solving Precision

From experiment of Figure 6.5, SRS is more precise than KM Center. The solutions occurs in top 10 list towards six problems.

Evaluation 3. system diagnosis and solution retrieving in time aspect.

In time evaluation, SRS system listed in Table 6.4 is more quick than expert.

Table 6.4

Time evaluation between SRS and KM Center

Evaluation Entities	Solution Retrieving Time (minutes)					
	db crash	missing redo log	archive log	data file	control file	system monitoring
SOS (min)	2.00	2.00	1.00	2.00	2.00	3.00
KM (min)	8.00	6.00	6.00	7.00	6.00	7.00

The diagram of time evaluation table (Table 6.4) is shown in Figure 6.6. Figure 6.6 shows the comparison result between SRS and KM Center towards system diagnosis and solution retrieving in time aspect.

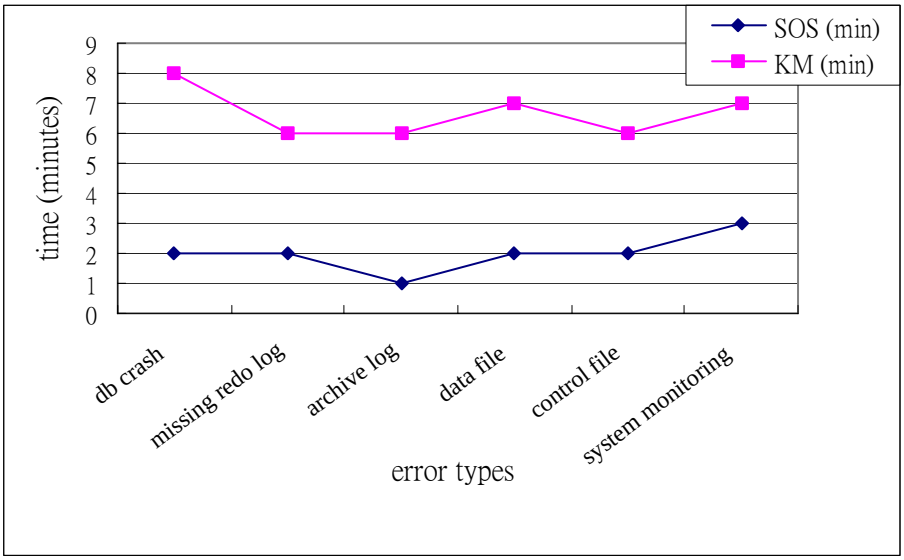


Figure 6.6. Time evaluation between SRS and expert

In time evaluation, SRS is quicker than KM Center in problem diagnosis and retrieving the corresponding solutions towards six kinds of error type. From experiment, the error type “db crash” is the complicated problem, and using KM approach needs eight minutes to do problem diagnosis and solution retrieving.

Chapter 7. Conclusion and Future Work

System diagnosis and solution retrieving are very important for IT employees, until now, most of them are unable to find a better way to solve problem, even though they have document center and search engine to assist employees to solve problems, but the cost is still too high and not efficient. Unlike traditional mechanism, SRS system is a hybrid system, using rule-based inference and case-based reasoning. It is proper for applying system diagnosis and solution retrieving. Thereby we can refine rule base and revise case base quickly, while new version created and new problem occurs. Besides, SRS system integrates with monitoring tool and mobility devices that make problem solving more efficient.

In this thesis, we design and implement Solution Offering System (SRS) to assist employee to solve problem and discover problem easily. Our main contributions are: (1) Provide hybrid architecture to solve system problem, use case-based inference to infer the error type and increase performance, and use case-based reasoning retrieving solved case to assist employee deal with the abnormal situations. (2) Define NORM structure of Oracle Application Server and Oracle database to enhance inference. (3) Define attributes in query and case to increase accuracy for finding solutions.

In future work, we wish to apply this hybrid architecture in different domain to help more employees, e.g., IC design and supply chain. Due to they have tree structure stuff on modules and the attributes could be identified and compared. Depending on tree structure, each rule class has its own objects and facts, the object name, fact name and fact value need to be modified to fulfill new related domains. On the other hand, attributes need to be modified in query and case to satisfy the similarity comparison in new domain. Basically, the architecture of SRS system could be kept for new related domains because this architecture imitates the thinking model from experts.

Bibliography

[1] Y. I. Chang and W. H. Hsieh, "**An efficient scheduling method for query-set-based broadcasting in mobile environments,**" Distributed Computing Systems Workshops, 2004. Proceedings. 24th International Conference, Page(s):478 – 483, 2004.

[2] Y. F. Chen, H. Huang, R. Jana, S. John, S. Jora, A. Reibman and B. Wei, "**Personalized multimedia services using a mobile service platform,"** Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE, Page(s):918 - 925 vol.2, 17-21 March 2002.

[3] Y. Chen, W. Y. Ma and H. J. Zhang, "**Detecting web page structure for adaptive viewing on small form factor devices,**" ACM TXN, 2003.

[4] M. Gu, X. Tong and A. Agnar, "**Comparing similarity calculation methods in conversational CBR, "Information reuse and integration**, Conf, 2005. IRI -2005 IEEE International Conference, Page(s):427 – 432, 15-17 Aug. 2005.

[5] M. J. Hajar and S. P. Lee, PhD, "**Applying machine learning using case-based reasoning (CBR) and rule-based reasoning (RBR) approaches to object-oriented application framework documentation**," Information Technology and Applications, 2005. ICITA 2005. Third International Conference Volume 1, Page(s):52 - 57 vol.1, 4-7 July 2005.

[6] S. Hayashi, T. Asakura and S. Zhang, "**Study of machine fault diagnosis system using neural networks,**" Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on Volume 1, Page(s):956 – 961, 12-17 May 2002.

[7] X. Hou, J. Gu, X. Shen and W. Yan, "**Application of data mining in fault diagnosis based on ontology**", Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05), 2005.

[8] S. Krishnaswamy, S. W. Loke and A. Zaslasvky "**A hybrid model for improving response time in distributed data mining,**" Systems, Man and Cybernetics, Part B, IEEE Transactions Volume 34, Issue 6, Page(s):2466 – 2479, Dec. 2004.

[9] F. R. Kumar, S. Gopalan and V. Sridhar, "**Context enabled Multi-CBR based Recommendation Engine for E-commerce**," e-Business Engineering, 2005. ICEBE 2005. IEEE International Conference, Page(s):237 – 244, 12-18 Oct. 2005.

[10] G. Lambert-Torres, H. G. Martins, R. Rossi and L. E. B. da Silva, "**Using similarity assessment in case-based reasoning to solve power system substation problems,**" Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference Volume 1, Page(s):343 - 346 vol.1, 4-7 May 2003.

[11] T. Lemlouma and N. Layaida, "**Adapted content delivery for different contexts,**" Applications and the Internet, 2003. Proceedings. 2003 Symposium, Page(s):190 - 197, 27-31 Jan. 2003.

[12] W. Y. Lum and F. C. M. Lau, "**User-centric adaptation of structured Web documents for small devices,**" Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference, Volume 1, Page(s):507 - 512 vol.1, 28-30 March 2005.

[13] W. C. Peng and M. S. Chen, "**Developing data allocation schemes by incremental mining of user moving patterns in a mobile computing system,**" IEEE Transactions on Knowledge and Data Engineering, Volume 15, Issue 1, Page(s):70 – 85, Jan.-Feb. 2003.

[14] I. Rish, M. Brodie, S. Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik and K. Hernandez, "**Adaptive diagnosis in distributed systems** ," IEEE Transactions on neural networkds, VOL. 16, NO. 5, SEPTEMBER 2005.

[15] Y. L. Wai and F. C. M. Lau, " **User-centric content negotiation for effective adaptation service in mobile computing,**" IEEE Transactions on Software Engineering, Page(s):1100 – 1111, Volume 29, Issue 12, Dec. 2003.

[16] D. Zhang, S. Dai, Y. Zheng, R. Zhang and P. Mu, "**Researches and application of a hybrid fault diagnosis expert system,**" Intelligent Control and Automation, 2000. Proceedings of the 3rd World Congress on Volume 1, Page(s):215 - 219 vol.1, 28 June-2 July 2000.

[17] Oracle Admin, "Oracle9i Application Server Architecture and Components," http://www.huihoo.com/oracle/application_server/9i.html.

[18] Oracle Admin, "Oracle to answer Regis MSCD 640 Oracle Admin on Windows 2000 Vocabulary Assignment Answers**,"** http://www.wilsonmar.com/1oraarch.htm.

[19] Oracle, OTN, http://www.oracle.com/technology/software/products/ias/index.html.

[20] Oracle, OTN, http://www.oracle.com/technology/software/products/database/oracle10g/index.html.

[21] Oracle, OTN, http://www.oracle.com/technology/software/products/jdev/index.html.

[22] 曾憲雄等, 人工智慧與專家系統—理論、實務、應用 ,2005, 旗標。

Appendix A: AT of Rule class Daemon home

	TROLL	TREC	CIP	CDFH	WLTD	CFDF
Memory loading	80%	85%	90%	85%	90%	95%
CPU loading	80%	85%	90%	85%	90%	95%
Transaction number	x>=1000	x>=1200	1<=x<=100	x>=1500	x>=500	x>=600
Not commit transaction	Yes	No	Slightly	Slightly	Slightly	Sometimes
Transaction abort	Yes	No	Slightly	Slightly	Sometimes	Sometimes
CKPT not trigger yet	No	Yes	Slightly	Slightly	Sometimes	Slightly
System pending	Very Often	Very Often	Sometimes	Often	Often	Often
Idle process number	1<= x <=10	1<= x <=10	x>=100	1<=x<=50	1<= x <=10	1<= x <=10
Disk Full	No	No	No	Slightly	Yes	Yes
Write back to control file	Slightly	Slightly	No	Slightly	Sometimes	Yes
Write back to date file	Slightly	Slightly	No	No	Sometimes	yes
Write back to log file	Slightly	Slightly	sometimes	No	Yes	sometimes

Where objects are listed as follows:

- TROLL: Transaction Rollback
- TREC: Transaction Recovery
- CIP: Check Idle ProcessesCDFH: Check data file header
- WLTD: Write log to db
- CFDF: Update control file and data file

Facts are listed as follows:

- Memory loading: The loading status of hardware memory is in DB.
- CPU loading: The loading status of CPU utility is in DB.
- Transaction number: The number of transaction processes in DB.
- Not Commit Transaction: The transactions those are modified but are not commit yet.
- Transaction abort: The transaction those are error or other related reason to trigger transaction abort.
- CKPT not trigger yet: The DB module “check point” is not been triggered yet.
- System Pending: The whole system (DB and Application server) are pending.
- Idle Process number: The processes number those are idle locate in current system.
- Disk Full: The volume of hardware disk that is full.
- Write back to control file: When system parameters are modified, the DB module will write modified information into control file.
- Write back to data file: When system data are modified, the DB module will write data into data files which are located in DB.
- Write back to log file: When system logs are modified and exceed the redo log buffer, the DB module will write logs into log files which are located in DB.

AOT of Rule class Daemon home

	TROLL	TREC	CIP	CDFH	WLTD	CFDF
Memory loading	3	3	3	3	4	3
CPU loading	3	2	3	3	3	2
Transaction number	3	3	3	2	3	3
Not commit transaction	D	X	1	2	1	1
Transaction abort	D	X	1	2	2	1
CKPT not trigger yet	2	D	2	1	1	1
System pending	D	3	1	2	2	3
Idle process number	1	2	D	2	1	2
Disk Full	3	3	3	3	3	4
Write back to control file	1	1	1	1	2	D
Write back to date file	1	1	1	1	2	D
Write back to log file	1	1	1	2	D	3

Where,

D: dominate the relationship

X: no relation

No. : 1~ 5 relationship strength

Appendix B: AT of Rule class DB Daemon

	System monitoring	Process monitoring	Check point	Log writer
Transaction rollback	no	yes	slightly	partial
Transaction recovery	no	yes	no	slightly
Clear idle process	partial	Yes	slightly	no
Clear data file header	no	No	slightly	no
Write log into disk	sometimes	sometimes	sometimes	yes
Update control file and data file	no	sometimes	yes	yes
Instance recovery	yes	no	no	no
Temp segment recovery	yes	no	no	no
Connect fragment	yes	no	no	no
Sub processes	1	1	1	1-10

Where objects are System monitoring, Process monitoring, Check point, and Log writer, please refer the Appendix C to reach the details.

The Facts are listed as follows:

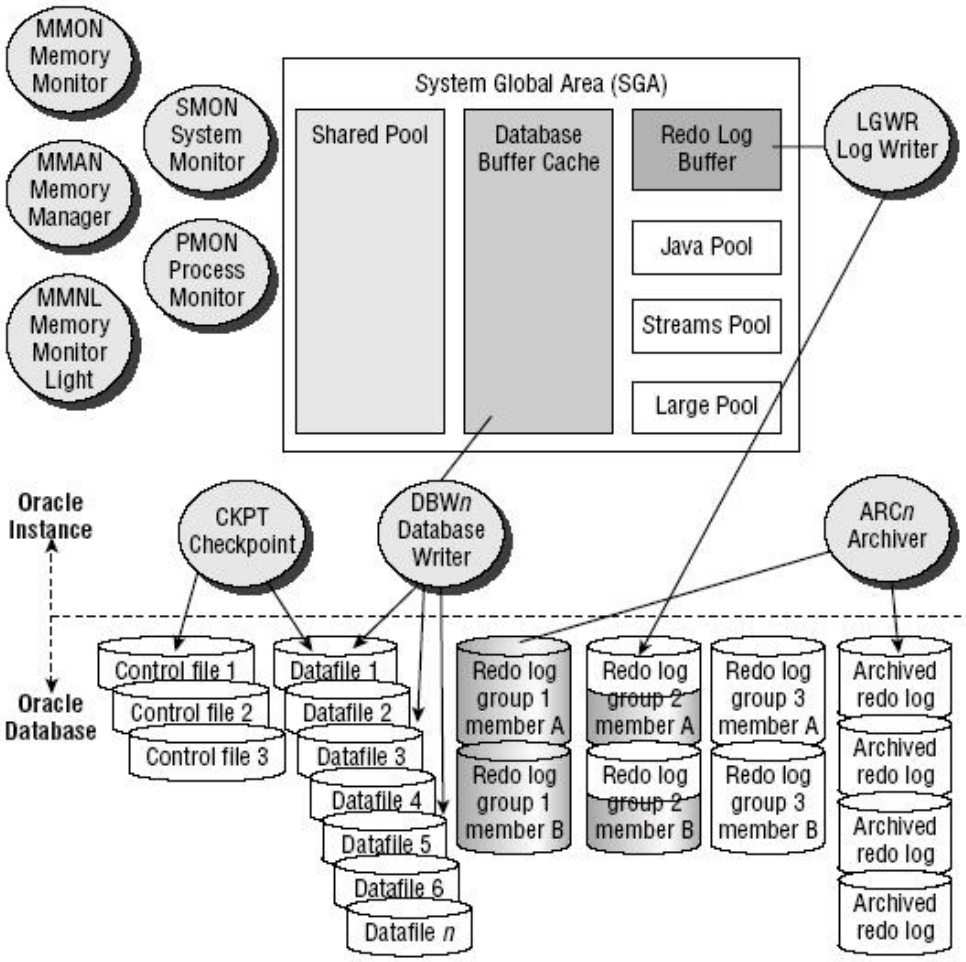
- Transaction rollback: When errors occurs in data saving, the related transactions will be roll back to their previous status.
- Transaction recovery: Transaction Recovery is an application recovery whereby the effects of specific transactions during a specified timeframe are removed from the database.
- Clear idle process: When idle processes exceed the numbers of system predefined setting, the DB module will clear idle processes.
- Clear data file header: When writing data file errors, the data file header will be clear by DB module.
- Write log into disk: When system produces some information, the DB module will write logs into disks.
- Update control file and data file: When system parameters or data have been modified, the DB module will update control file and data file which are located in DB.
- Instance recovery: Occurs when a software or hardware problem prevents an instance from continuing work.
- Temp segment recovery: Occurs when a segment space is not enough or software error.
- Connect fragment: When system fragments are exceeding a value, then the DB module will connect the fragment block into a larger block for DB reuse.
- Sub process: Sub process is the number of processes fork by main process.

AOT of Rule class DB Daemon

	System monitoring	Process monitoring	Check point	Log writer
Transaction rollback	2	D	1	3
Transaction recovery	2	D	X	1
Clear idle process	3	D	2	X
Clear data file header	X	X	1	X
Write log into disk	2	2	2	D
Update control file and datafile	1	2	D	D
Instance recovery	D	0	2	1
Temp segment recovery	D	0	1	1
Connect fragment	D	0	0	0
Sub processes	3	3	2	2

Where,
D: dominate the relationship
X: no relation
No. : 1~ 5 relationship strength

Appendix C: Oracle database 10g architecture



Components:

SGA are made up three require components and three optional components:

- Shared Pool – Cache the most recently used SQL statements that have been issued by database users.
- Database buffer Cache—Cache the data that has been recently accessed by database users.
- Redo log buffer—Store transaction information for recovery purpose.
- Java pool—Caches the most recently used Java objects and application code when Oracle's JVM option is used.
- Large pool—Cache data for large operations such as Recovery Manager backup and restore activity and Shared Server components.
- Stream pool—Cache the data associated with queued message requests when Oracle's Advanced Queuing option is used.

Oracle background process

- Memory Monitor (MMON)—Gather and analyze statistics used by Automatic Workload

Repository.

- Memory Manager (MMAN)—Manages the individual SGA component when Automatic Shared Memory Management feature is used.
- Memory Monitor Light (MMNL)--Gather and analyze statistics used by Automatic Workload Repository feature, Flushed every 30 minutes or when buffer is full.
- System monitor (SMON)—Performs instance recovery following an instance crash, combine free spaces in database, and managed spaces used for sorting.
- Process Monitor (PMON)—Clean up failed database connections.
- Database Writer (DBWn)—Writes modified db blocks from the SGA's db buffer cache to the datafiles.
- Log Writer (LGWR)—Writes transaction recovery information from the SGA's Redo Log Buffer to the online redo log file.
- Checkpoint (CKPT)—Update the database files following a Checkpoint Event.
- Archiver (Arcn)—archive redo log in second place for recovery

Characteristic:

- 10g RAC—clusters and fail-over recovery mechanism.
- Oracle HTML DB--A secure, web-based, metadata-driven, database-centric application development and deployment platform.
- Oracle vs. Sybase—
  - Oracle DB has raw partitions, but Sybase.
  - Oracle DB can be implemented on different platform, Sybase also can do that but not very stable.
  - Performance is better than Sybase.
  - Oracle use multi-layers network calculation can use OCI, ODBC and JDBC to connect with clients.
  - Sybase--C/S architecture ODBC, Jconnect and Ct-library connect with clients.
  - Both provide GUI and command line, Oracle use both on solaris and windows, Sybase's GUI is often unable to show the current status.

Oracle 10g DB have RAC, online html DB and good association with application server, and provide multiple compatible platforms (Solaris, Linux and Windows) hence that is compatible to develop ERP, and large system for Enterprise.

Autobiography

Please be informed that Tsung-Ping is a professional person with good communication skills and high EQ in his stuff. Familiar with various integration solutions, he dedicates in various solutions (Expert System, Business Intelligent, Oracle DB and Oracle Application Server, etc). Not only technique skills but also good working spirit in team work. In fact, he is a workaholic on technique research and integrations.

Tsung-Ping loves to play golf, go swimming, and mountain climbing. He is also a good guitar player, related in practicing pop music and classical music over 10 years.

He is a person with humor sense, simultaneously he makes a lot of fun to colleagues and friends around him. Never give up is his personal characteristic. He is always full of energy and passion on his job.

Tsung-Ping studied in Computer Science in Feng-Chia University between Sep, 1993 to June 1997 and got bachelor degree in June 1997. At the same time, Tsung-Ping studied in Degree Program of Electrical Engineering and Computer Science College of Computer Science in National Chiao Tung University between June 2004 to June 2006 and got master degree in July 2006.