VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

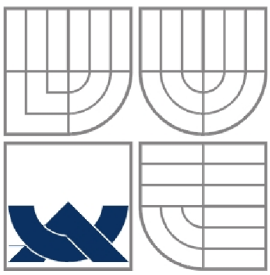
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTROJE NA PODPORU ADMINISTRACE A LADĚNÍ VÝKONNOSTI DATABÁZOVÉHO SERVERU ORACLE 11G

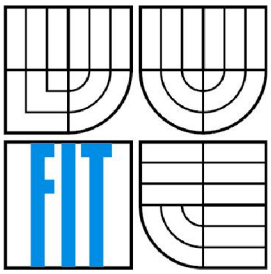
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE VOJTĚCH FRÖML
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTROJE NA PODPORU ADMINISTRACE A LADĚNÍ VÝKONNOSTI DATABÁZOVÉHO SERVERU ORACLE 11G

ADMINISTRATOR TOOLS FOR ORACLE 11G

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — VOJTĚCH FRÖML
AUTHOR

VEDOUCÍ PRÁCE — Doc. Ing. JAROSLAV ZENDULKA, CSc.
SUPERVISOR

BRNO 2010

Abstrakt

Tato bakalářská práce představuje koncepty správy databázového systému Oracle 11g. Podává přehled nástrojů a technologií pro správu, monitorování a ladění tohoto systému. Konkrétněji se zaměřuje na možnosti v oblasti ladění příkazů jazyka SQL. Dále popisuje návrh a implementaci ukázkové aplikace SQLTuningTool. Tato aplikace vytváří SQL zátěž databáze simulací klientského prostředí a prezentuje některé databázové nástroje pro monitorování SQL příkazů a jejich ladění. Závěrem jsou kriticky zhodnoceny dosažené výsledky a nastíněny možnosti dalšího pokračování projektu.

Abstract

This bachelor thesis presents administration concepts of Oracle Database 11g. It provides overview of administrative, monitoring and tuning tools and technologies. Options in the area of tuning SQL statements are examined in greater detail. Furthemore it describes design and implementation of example application SQLTuningTool. This application creates database SQL workload and presents some of the database tools for monitoring and tuning SQL statements. The end section discusses results and proposes options for further continuation of this project.

Klíčová slova

Oracle Database 11g, technologie Oracle, Enterprise Manager, správa databáze, monitorování databáze, ladění SQL

Keywords

Oracle Database 11g, Oracle technologies, Enterprise Manager, database administration, database monitoring, SQL tuning

Citace

Vojtěch Fröml: Nástroje na podporu administrace a ladění databázového serveru Oracle 11g, bakalářská práce, Brno, FIT VUT v Brně, 2010

Nástroje na podporu administrace a ladění databázového serveru Oracle 11g

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Doc. Ing. Jaroslava Zendulky, CSc.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Vojtěch Fröml
18. května 2010

Poděkování

Rád bych poděkoval Doc. Ing. Jaroslavu Zendulkovi, Csc. za jeho doporučení, rady a čas, které mi věnoval.

© Vojtěch Fröml, 2010
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Obsah....1
1 Úvod....3
2 Přehled nástrojů a technologií....5
2.1 Licencovatelné balíčky k Oracle 11g....5
2.2 Balíčky správy databáze....7
3 Oracle Enterprise Manager 11g....9
3.1 Database Control....10
3.2 Grid Control....11
4 Správa a ladění serveru Oracle 11g....12
4.1 Nástroje pro monitorování systému....12
4.1.1 Pohledy datového slovníku....12
4.1.2 Dynamické pohledy....14
4.1.3 Automatic Workload Repository....17
4.1.4 Automatic Database Diagnostic Monitoring....19
4.2 Nástroje pro ladění výkonnosti systému....22
4.2.1 Úvod do ladění příkazů jazyka SQL....23
4.2.2 Prováděcí plány příkazů SQL....24
4.2.3 SQL Tuning Advisor....25
4.2.4 Automatizace ladění příkazů SQL....26
4.2.5 Monitorování SQL v reálném čase....27
4.3 Nástroje pro správu systému....28
4.3.1 Správa diskového místa....29
4.3.2 Správa paměti....31
5 Návrh aplikace....32
5.1 Úvod do projektu....32
5.2 Simulace SQL zátěže....32
5.3 Monitorování zátěže....33
5.4 Ladění SQL příkazů....34
5.5 Návrh tříd....36
6 Implementace....37
6.1 Nástroje použité pro realizaci projektu....37
6.2 Uživatelské rozhraní....37
6.3 Databázové schéma....38
6.4 Klientské prostředí....38

6.5 Generování SQL příkazů....39
6.6 Administrátorské prostředí....40
7 Závěr....43
Literatura....44
Seznam příloh....46
Příloha A....47
Příloha B....50

1 Úvod

Efektivita moderních relačních databázových systémů (RDBMS – *Relational Database Management System*) má přímý vliv na výši nákladů spojených s prováděním každodenních administrativních úkolů. Lze ji v komparativně vyhodnotit v několika oblastech správy databáze [1]:

- dostupnost
- škálovatelnost
- bezpečnost
- spravovatelnost
- vývoj aplikací
- skladování dat

Typickým požadavkem na databázový systém ze strany provozovatele je schopnost zajistit dostatečnou, ideálně stoprocentní dostupnost databáze. Ke splnění tohoto cíle je třeba úplně zamezit či minimalizovat negativní dopady výpadků způsobenými různými plánovanými i nečekanými okolnostmi, včetně chyby lidského faktoru. Spolu s kvalitními administrativními možnostmi je tedy nezbytná implementace samo-spravovacích automatických nástrojů.

Dalším faktorem k zvýšení dostupnosti je vhodná decentralizovanost a škálovatelnost databázového systému. Např. rozložení zátěže mezi více databázovými servery poskytuje požadovanou pružnost při nutnosti zvýšení výkonu systému či navýšení kapacit datových úložišť.

Efektivitu aplikačního prostředí z pohledu správce databáze zásadně zvyšuje ovladatelnost systému. Tu představuje administrativní rozhraní, zjednodušující komplexnost prováděných spravovacích úloh, a to jak reaktivních (řešení nastalé situace) či proaktivních (monitorování systému a prevence).

Pro vývoj kvalitních klientských aplikací musí být v databázovým systémem integrována podpora efektivních dotazovacích jazyků, nástrojů a řešení konektivity.

Dále jsou nezbytné nástroje pro správu databázové architektury, fyzických a logických úložných datových struktur a správu paměti.

Databázový systém Oracle 11g, vyvinut firmou Oracle Corporation, je multiplatformní, otevřený, komplexní databázový systém poskytující rozsáhlou sadu pokročilých nástrojů ve všech těchto oblastech [2]. Cílem této práce je selektivní představení nástrojů pro správu tohoto databázového systému a seznámení se s jeho administrativními technologiemi s důrazem na nově představené v této verzi a nástroje pro ladění výkonu databáze. Nejedná se o kompletní přehled možností systému Oracle 11g, vynechány jsou např. kapitoly o bezpečnosti, vývoji aplikací zálohování a obnově či *Oracle Net*.

Oracle Database 11g je dostupná v několika typech instalací: *Express*, *Personal*, *Standard One*, *Standard* a *Enterprise*. Edice Express je omezena 1GB operační paměti a 4GB diskového prostoru. Vydání *Personal* nelze používat v produkčním prostředí, edice *Standard One* se od

Standard liší tím, že nepodporuje technologii Real Application Clusters. Verze *Enterprise* poskytuje nejširší paletu funkcí a možnost licencovat dodatečné balíčky zlepšující výkon, škálovatelnost, dostupnost, bezpečnost a ovladatelnost. Všechny vydání poskytují kromě podpory pro platformy *Windows*, *Linux* a *Unix* i podporu 64bit [2]. Tato práce se věnuje možnostem edice *Enterprise* včetně dodatečných balíčků.

Kapitola 2 seznamuje s některými pokročilými nástroji a technologiemi *Oracle 11g* poskytovanými dodatečnými balíčky včetně klíčových nástrojů, které byly představeny v nejnovější verzi. V kapitole 3 je popsáno uživatelské rozhraní *Oracle Enterprise Manager 11g*, které zajišťuje ovladatelnost a správu databáze Oracle. Konkrétní nástroje pro správu, monitorování a ladění výkonnosti popisuje kapitola 4. Návrh a implementaci aplikace nazvané *SQLTuningTool* prezentuje kapitola 5, respektive 6. Poslední kapitolou je závěr s návrhy na pokračování tohoto projektu.

2 Přehled nástrojů a technologií

U technologií uvedených v této kapitole je uveden pouze jejich účel a základní funkčnost, detailnější popis nástrojů pro správu, monitorování a ladění databázového systému Oracle 11g se nacházejí v kapitolách 3 a 4.

2.1 Licencovatelné balíčky k Oracle 11g

Tato kapitola představuje nástroje a technologie, které poskytují licencovatelné dodatečné balíčky nad rámec standardních funkcí edice *Oracle 11g Enterprise*. Popis nástrojů v této sekci vychází z oficiální stránky Oracle [2] a dokumentu [3].

- **Oracle Active Data Guard**

Oracle Active Data Guard zlepšuje faktor *Quality of Service* přesunutím zátěže na jednu či více synchronizovaných záložních nečinných databází. Povoluje čtení z této databáze za současného aktualizování změn v produkční databázi. Pro testovací účely může *Oracle Active Data Guard* záložní databázi otevřít po skončení úlohy opět uzavřít pro účel nutnosti obnovení zálohy v případě selhání produkční databáze

- **Oracle Advanced Compression**

Oracle Advanced Compression efektivně komprimuje veškerá strukturovaná i nestrukturovaná data, jako jsou např. dokumenty, obrázky, multimédia, provoz na síti či právě zálohovaná data.

- **Oracle Advanced Security**

Oracle Advanced Security umožňuje kódování dat uložených v databázi a kódování síťového přenosu dat dle průmyslových standardů. Podporuje výběr algoritmů a síly šifer.

- **Oracle Database Vault**

Oracle Database Vault řídí kontrolu uživatelských přístupů k aplikacím a datům. Protože nejčastější bezpečnostní hrozbou jsou nedůvěryhodní vnitřní uživatelé databáze, je žádoucí, aby některým uživatelům či správcům databáze byl odmítnut přístup ke specifickým datům na základě jejich oprávnění či denní hodiny.

- **Oracle Data Mining**

Oracle Data Mining umožňuje zákazníkům nacházet v jejich datech pomocí prediktivních analýz skryté vzory. Vývojáři aplikací mohou tyto nálezy automatizovat a distribuovat je v reálném čase.

- **Oracle In-Memory Database Cache**

Oracle In-Memory Database Cache zrychluje transakční dobu odezvy ukládáním nejvíce vytížených částí databáze na aplikační vrstvu. Kozistenci dat v této cache pak zajišťuje automatickým aktualizováním. Dotazování vyrovnávácí paměti umožňuje aplikaci přistupovat či měnit data mnohem rychleji.

- **Oracle Label Security**

Oracle Label Security přidává další stupeň bezpečnosti pro citlivá data, poskytuje mnohoúrovňové bezpečnostní funkce k ochraně dat až na úroveň jednotlivých řádků v tabulce.

- **Oracle OLAP**

Oracle OLAP je server pro on-line analytické zpracovávání dat vestavěný v databázi Oracle. Optimalizátor databáze Oracle může přesměrovat SQL dotazy na úroveň *OLAP kostky*. Ve verzi 11g je nově implementována podpora vysoce komprimovaných *OLAP materializovaných pohledů*, které zjednodušují správu dat a výrazně zrychlují transakční odezvu.

- **Oracle Partitioning**

Oracle Partitioning umožňuje fyzické rozdělení příliš velké tabulky při současném zachování aplikačního jednotabulkového pohledu. Zlepšuje ovladatelnost velkých tabulek a jejich indexů, prostřednictvím optimalizátoru dotazů následně i výkon databáze.

- **Oracle Real Application Clusters**

Oracle Real Application Clusters, v dřívějších verzích databáze Oracle pod názvem *Oracle Parallel Server*, umožňuje více instancím na oddělených serverech přístup ke stejným databázovým souborům. Spojuje dohromady technologii *Oracle Managed Files*, tabulkové prostory typu *bigfile*, robustní síťovou architekturu a technologii *Advanced Storage Management* [4]. Prostředí *RAC* rozkládá zátěž na více serverů, selhání jednoho uzlu tedy neovlivní relace klientů, případně dostupnost celého clusteru. RAC je také poskytováno v jednouzlové verzi *One Node*. V případě selhání je instance Oracle restartována na volném serveru v clusteru, následuje přesměrování klienta.

- **Oracle Real Application Testing**

Oracle Real Application Testing umožňuje firmám bez rizika ztráty dat testovat nové technologie. Kombinuje monitorování zátěže, zálohování a SQL analyzátor výkonu, je tedy možné testovat nové prostředí oproti reálné zátěži a před nasazením do produkce jej optimalizovat.

- **Oracle Spatial**

Oracle Spatial integruje do databázového prostředí podporu pro optimalizované ukládání prostorových dat. Poskytuje analýzu takto uložených dat na základě jejich prostorových charakteristik [5].

- **Oracle Total Recall**

Oracle Total Recall poskytuje mechanismus, jak bezchybně archivovat veškeré minulé změny prováděné nad databází Oracle. Takto archivovaná data mohou být zpřístupněna pomocí Flashback Query. Na základě specifikovaných pravidel je tato databáze pravidelně promazávána.

2.2 Balíčky správy databáze

Dodatečné balíčky *Database Management Packs* obsahují nástroje pro diagnostiku, konfiguraci a optimalizaci databázového systému Oracle 11g přístupné přes webové rozhraní *Oracle Enterprise Manager*. Popisy těchto balíčků vycházejí z oficiálních stránek Oracle [6].

- **Oracle Change Management Pack**

Oracle Change Management Pack umožňuje administrátorům databáze bezpečně provádět komplexní změny databázových objektů, sledovat změny nad schematy a databázemi, kopírovat schemata a objekty a porovnávat a synchronizovat schemata a databáze.

- **Oracle Configuration Management Pack**

Oracle Configuration Management Pack umožňuje sledování hardwarových a softwarových konfiguračních informací pro hostitelské počítače a databáze spravované nástrojem Enterprise Manager a manipulaci s těmito daty. Klíčová výhoda tohoto balíčku je zpřístupnění možnosti urychlení řešení problémů porovnáním konfigurací dvou databází.

- **Oracle Data Masking Pack**

Oracle Data Masking Pack umožňuje organizacím bezpečné sdílení produkčních dat s vývojáři aplikací v testovacích prostředích bez porušení soukromí či pravidel důvěrnosti.

- **Oracle Diagnostic Pack**

Oracle Diagnostic Pack poskytuje pokročilou automatickou diagnostiku a monitorování výkonu databáze. Integruje úložiště *Automatic Workload Repository* (AWR) pro kolekci statistických údajů, nástroj *Automatic Database Diagnostic Monitor* (ADDM) pro generování statistických sestav, monitorování výkonu *Performance Monitoring*, upozornění *Event notifications*, historie událostí *Event History* a nastavení výpadku *Blackout*. Tento balíček je podrobněji rozebrán v kapitole 4.

- **Oracle Provisioning and Patch Automation Pack**

Oracle Provisioning and Patch Automation Pack automatizuje aplikaci softwaru, aplikací a oprav. Poskytuje automatické patchování produktů Oracle a operačního systému, databáze *Critical Patch Facility* a *Real Application Clusters*, dále poskytuje různé aplikace a instanci pro konverzi *RAC* na jedno kliknutí.

- **Oracle Tuning Pack**

Oracle Tuning Pack poskytuje administrátorům databáze nástroje pro pokročilé ladění výkonu prostředí Oracle, včetně ladění příkazů jazyka SQL *SQL Tuning*. Nástroje v tomto balíčku jsou podrobněji rozebrány v kapitole 4 a kapitolách věnujících se návrhu a implementace vlastní aplikace.

3 Oracle Enterprise Manager 11g

Oracle Enterprise Manager (OEM) je webové ovládací rozhraní implementující v sobě nástroje, které umožňují správu všech komponent infrastruktury Oracle, včetně instancí databáze Oracle, aplikačních serverů Oracle a webových serverů. Existují-li pro aplikace třetích stran agenti pro správu, může OEM spravovat i tyto aplikace stejným způsobem jako ostatní komponenty Oracle [4].

Základním účelem nové verze 11g, vydané 22. dubna 2010, je co největší podpora konceptu *Business Driven Application Management* [7], tedy maximálnímu přizpůsobení IT aplikací reálným potřebám firmy. Byla rozšířena paleta nástrojů OEM 10g pro sběr informací o tom, jakým způsobem používají zákazníci aplikaci a jak splňuje jejich požadavky, což umožňuje IT oddělením lépe optimalizovat návrh a implementací aplikace.

OEM 11g byl představen s konceptem integrovaného ovládání všech komponent databázového systému (obr. 3.1).

Obrázek 3.1: Schéma integrovaného ovládání OEM 11g (*Převzato z* [8])

Celý databázový systém je od aplikační vrstvy po fyzické úložiště na disku ovládán z jedné konzoly. Tato architektura umožňuje odstranění nadbytečných správních IT úkonů často prováděných různými stranami a v důsledku tedy šetří zbytečné náklady na správu databázového systému.

Posledním klíčovým konceptem verze 11g je integrace systému ovládání a technické podpory [7]. Při nastalém problému se administrátor databáze tradičně snažil diagnostikovat a vyřešit situaci sám, eventuálně hledal řešení na technické podpoře odpovídající problémovému produktu. OEM 11g obsahuje nové nástroje pro sběr kompletních konfiguračních statistik. Systém *Integrated Systems Management and Support* umožňuje administrátorům použít nástroje pro automatickou diagnostiku systému a využít konfigurační statistiky k získání on-line podpory *My Oracle Support* pro stažení řešení či softwarové opravy.

OEM 11g existuje ve dvou verzích – *OEM Database Control* pro decentralizovanou správu jedné databázové instance ovládací konzolou a *OEM Grid Control* jako centralizované řešení ovládání a automatizace mřížky (*grid*) Oracle technologií i softwaru třetích stran.

3.1 Database Control

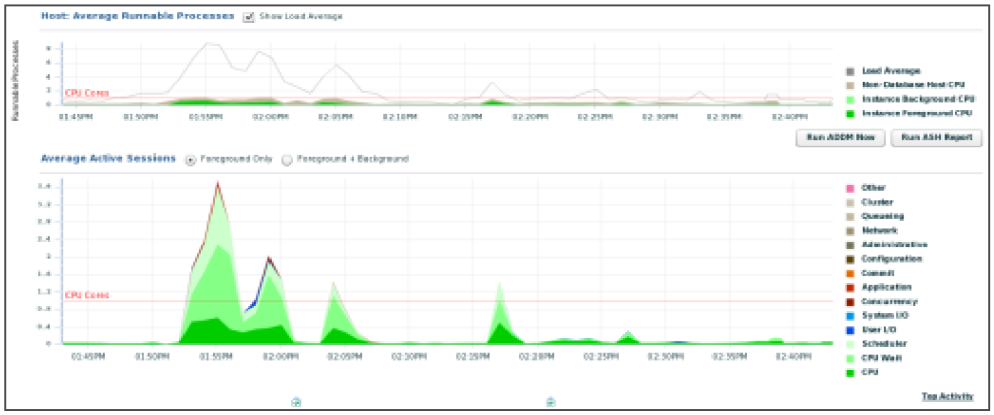
OEM Database Control byl představen ve verzi 10g jako nástupce Java klienta schopného ovládat a konfigurovat databáze. Licencované balíčky popsané výše jsou díky OEM Database control užitné okamžitě po instalaci v menu *Server*.

Obrázek 3.2: OEM Database Control menu

Úvodní stránka obsahuje základní informace o vytížení hostitelského systému (obr 3.3)

Obrázek 3.3: OEM Database Control úvodní obrazovka

Obrazovka *Performance* poskytuje výstupy monitorování databáze v reálném čase pomocí výchozí sady statistických dat SYSTEM_MOVING_WINDOW (obr 3.4), viz kapitola 4.



Obrázek 3.4: OEM Database Control monitorování výkonu

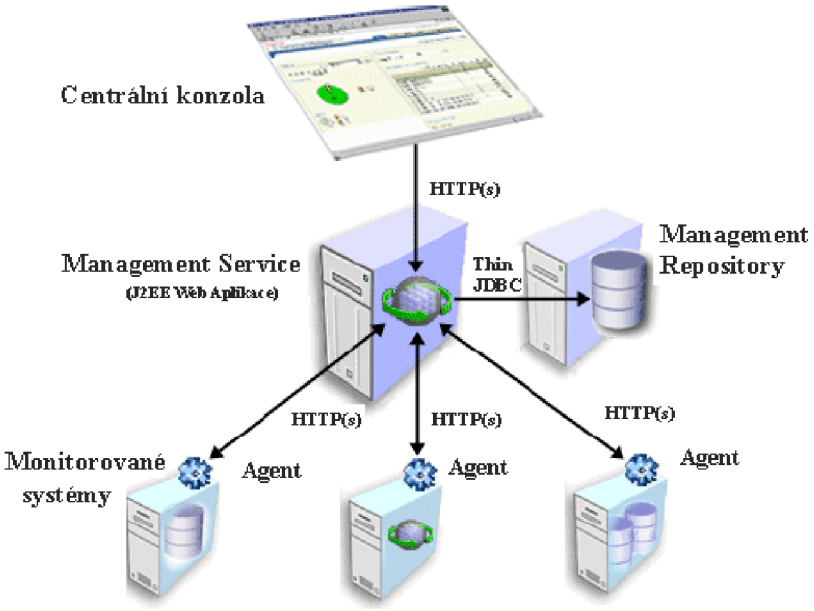
3.2 Grid Control

OEM Grid Control slouží k hromadnému spravování databází a aplikačních serverů Oracle. Poskytuje také možnosti rozšíření platform vyvinutých třetími stranami, jako např. IBM DB2 Database, Microsoft SQL Server, Microsoft .NET či NetApp Filter [9]. Architektura OEM Grid Control se skládá ze třech částí [10] (obr. 3.5):

- Oracle Management Agent (OMA)
- Oracle Management Server (OMS)
- Oracle Management Repository (OMR)

OMA je spuštěn na koncovém uzlu a slouží pro sběr a generování informací o hardwaru, operačním systému a spuštěných aplikacích. Pomocí instalovaných pluginů na každém OMA lze rozšířit rozsah informací prezentovaných serveru.

OMS běží na jednom či více serverech, sbírá informace poskytované OMA a hromadně je ukládá do úložiště OMR.



Obrázek 3.5: Schéma OEM Grid Control (*Převzato z* [16])

4 Správa a ladění serveru Oracle 11g

Monitorování výkonu databáze, dohled nad jejím optimálním chodem a implementace korektivních opatření jsou důležitými úkoly administrátora databáze. Tato kapitola popisuje nejvýznamější nástroje, které k těmto účelům poskytuje databázový systém Oracle 11g. Pokud není uvedeno jinak, informace v této kapitole včetně několika ilustrací jsou čerpány z oficiální dokumentace Oracle 11g R2 [11].

4.1 Nástroje pro monitorování systému

Základní nástroj pro monitorování systému představují **pohledy datového slovníku** a **dynamické pohledy**. Pokročilá sada nástrojů integrovaná v jádře databázového systému a v grafickém prostředí OEM [12] je dodávána v dodatečném balíčku *Oracle Diagnostic Pack*, a je nutnou prerekvizitou pro pro užívání ladicích nástrojů v dodatečném balíčku *Oracle Tuning Pack*.

Diagnostický balíček nabízí funkce monitorování a automatizované diagnostiky databázového výkonu. Pravidelné shromažďování detailních informací o zátěži a databázových aktivitách zajišťuje úložiště statistických dat **Automatic Workload Repository**.

4.1.1 Pohledy datového slovníku

Důležitá data o současném stavu využití diskového prostoru poskytují pohledy datového slovníku. Tabulky, nad kterými jsou vytvořeny začínají prefixem `_DBA` a zapouzdřují nejdůležitější metadata Oracle s detaily o tabulkách, indexech, paměti SGA a fyzických úložištích. Aktualizují se pouze při změnách datového slovníku, obvykle tedy ne příliš často, a mají převážně statický charakter. Mnoho z těchto pohledů má svůj ekvivalent s prefixem `_USER`, který poskytuje informace pouze o aktuálním uživateli (schematu).

Přehled několika důležitých pohledů datového slovníku:

- `DBA_USERS` – poskytuje informace o všech uživatelech databáze

  - následující příkaz vybere aktuálně otevřené databázové účty

```
SQL> select user_id, username, default_tablespace from
     dba_users where account_status = 'OPEN';
```

- `DBA_TABLESPACES` – poskytuje informace o všech tabulkových prostorech databázemi

  - následujícím příkazem získá informace o typu tabulkových prostorů

```
SQL> select tablespace_name, block_size, contents, from
     dba_tablespaces;
```

- DBA_TABLES – poskytuje informace o tabulkách

  - následujícím příkazem je možno získat informace o typu tabulkových prostorů

```
SQL> select * from dba_tables
     where tablespace_name = 'EXAMPLE';
```

- DBA_INDEXES – poskytuje informace o tabulkových indexech, pomáhá při rozhodování, zda je index užitečný či ne. Příbuzný pohled DBA_IND_COLUMNS ukazuje které sloupce tabulky jsou indexované

  - následujícím příkazem je možno získat seznam indexů

```
SQL> select index_name, index_type, table_name
     from dba_indexes group by table_name;
```

- DBA_DATA_FILES – poskytuje informace o datových souborech

  - následujícím příkazem seřadí databázové soubory podle velikosti

```
SQL> select file_name, tablespace_name, bytes, blocks
     from dba_data_files order by bytes desc;
```

- DBA_SEGMENTS – obsahuje záznam pro každý segment databáze

  - následujícím příkaz zjistí počet bajtů, segmentů a rozsahů všech tabulkových prostorů

```
SQL> select tablespace_name, sum(bytes), sum(blocks),
     sum(extents) from dba_segments
     group by (tablespace_name);
```

- DBA_EXTENTS – obsahuje záznamy pro rozsahy (extents) databáze včetně podrobných informací o každém databázovém objektu

  - následujícím příkaz zjistí velikost rozsahů a soubor, ve kterém jsou uloženy

```
SQL> select owner, segment_name, tablespace, extent_id,
     file_id, block_id, bytes from dba_extents
     where segment_name = 'SALES';
```

- DBA_UNDO_EXTENTS – obsahuje záznamy o rozsazích používaných nástrojem *Automatic Undo Management*

- DBA_FREE_SPACE – obsahuje data tabulkového prostoru rozdělená podle čísle datových souborů

  - typickým použitím je spočítání volného místa v tabulkovém prostoru následujícím příkazem

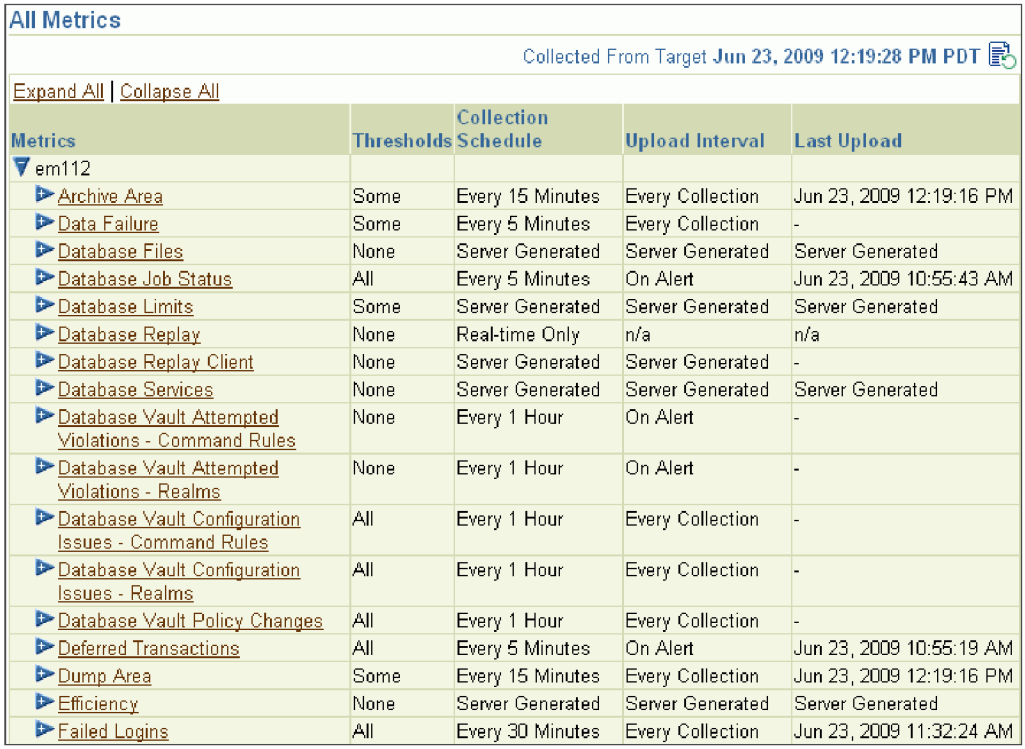
```
SQL> select tablespace_name, sum(bytes) from dba_free_space
     group by tablespace_name;
```

- DBA_THRESHOLDS – poskytuje prahové metriky obsahující podmínky, za kterých je v případě jejich překročení aktuálním stavem databáze generován poplach (*alert*) (viz. kapitola 4.1.2). Pomocí rozhraní OEM lze nastavit automatické zasílání informací o poplaších

elektronickou poštou či na pager. Po instalaci databázového systému Oracle 11g jsou implicitně nastaveny např. tyto hodnoty [4]:

- Minimálně jedna relace uživatele je alespoň tři minuty blokována
- Celkový počet souběžně běžících procesů dosáhl 80 procent hodnoty inicializačního parametru PROCESSES
- Celkový počet uživatelských relací dosáhl 80 procent hodnoty inicializačního parametru SESSIONS
- Více než 1200 současně otevřených kurzorů
- Více než 100 přihlášení za sekundu
- Tabulkový prostor je z více než 85% procent zaplněn (varování) nebo je zaplněn z více než 95% (kritická zpráva)

V nástroji OEM Database Control lze nalézt zobrazení právě používaných metrik v sekci *Related Links* na hlavní stránce pod odkazem **All Metrics** (obr 4.1).



All Metrics

Collected From Target **Jun 23, 2009 12:19:28 PM PDT**

Expand All | Collapse All

Metrics	Thresholds	Collection Schedule	Upload Interval	Last Upload
em112				
Archive Area	Some	Every 15 Minutes	Every Collection	Jun 23, 2009 12:19:16 PM
Data Failure	Some	Every 5 Minutes	Every Collection	-
Database Files	None	Server Generated	Server Generated	Server Generated
Database Job Status	All	Every 5 Minutes	On Alert	Jun 23, 2009 10:55:43 AM
Database Limits	Some	Server Generated	Server Generated	Server Generated
Database Replay	None	Real-time Only	n/a	n/a
Database Replay Client	None	Server Generated	Server Generated	-
Database Services	None	Server Generated	Server Generated	Server Generated
Database Vault Attempted Violations - Command Rules	None	Every 1 Hour	On Alert	-
Database Vault Attempted Violations - Realms	None	Every 1 Hour	On Alert	-
Database Vault Configuration Issues - Command Rules	All	Every 1 Hour	Every Collection	-
Database Vault Configuration Issues - Realms	All	Every 1 Hour	Every Collection	-
Database Vault Policy Changes	All	Every 1 Hour	Every Collection	-
Deferred Transactions	All	Every 5 Minutes	On Alert	Jun 23, 2009 10:55:19 AM
Dump Area	Some	Every 15 Minutes	Every Collection	Jun 23, 2009 12:19:16 PM
Efficiency	None	Server Generated	Server Generated	Server Generated
Failed Logins	All	Every 30 Minutes	Every Collection	Jun 23, 2009 11:32:24 AM

Obrázek 4.1: Zobrazení metrik v OEM Database Control

4.1.2 Dynamické pohledy

Databáze Oracle si za běhu vytváří **dynamické tabulky výkonnosti**, což je kolekce virtuálních tabulek začínajících prefixem `V_$`, které zaznamenávají aktivitu nad databází. Tyto objekty nejsou tabulkami v pravém slova smyslu a neměli by k nim mít přístup běžní uživatelé. Nad dynamickými tabulkami jsou vytvořeny pohledy, které mají veřejná jména uvozena prefixem `V$`.

Přehled několika důležitých dynamických **pohledů:**

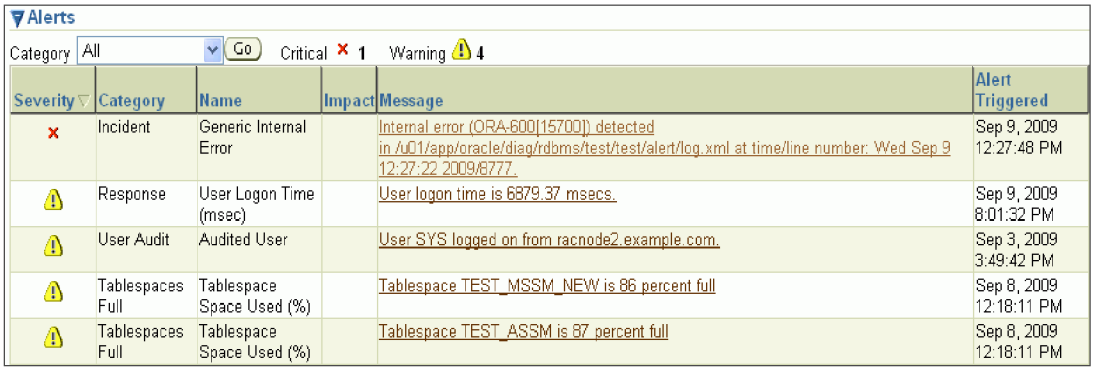
- `V$ACTIVE_SESSION_HISTORY` – zobrazuje navzorkovanou statistiku aktivních databázových relací, sbíranou v intervalu 1 minuty. Pohled obsahuje jeden řádek pro každou aktivní relaci ve vzorku. Sběr těchto statistik zajišťuje nástroj *Automatic Workload Repository*, implicitně probíhá v intervalu 1 hodiny. Formát dat poskytovaných tímto pohledem je podobný pohledu `V$SESSION`

  Ve verzi Oracle Database 11g Release 2 byla funknčnost této klíčové komponenty pro monitorování zátěže rozšířena o shromažďování dodatečných RAC informací týkajících se událostí čekání (*wait events*) nejvíce ovlivňujících výkon clusteru. Dále byla funkce ASH rozšířena o možnost monitorování nečinných databází pro analýzu výkonu nástroje DataGuard [12].

- `V$ALERT_TYPES` – obsahuje seznam podmínek pro vznik poplachů, které je možné sledovat. Např. poplachy problémů se správou místa lze nalézt ve skupině s názvem „Space“ - následujícím příkazem lze zjistit poplachy skupiny space

```
SQL> select reason_id, object_type, scope,
     internal_metric_category, internal metric_name
     from v$alert_types where group_name = 'Space';
```

  V nástroji OEM Database Control lze nalézt vyvolané poplachy pod hlavičkou **Alerts** na hlavní obrazovce (obr. 4.2).



Obrázek 4.2: Zobrazení vyvolaných poplachů v OEM Database Control

- `V$DATABASE` – obsahuje informace o databázi uvedené v řídícím souboru
- `V$DATAFILE` – obsahuje informace o datových souborech uvedené v řídícím souboru
- `V$FIXED_TABLE` – obsahuje seznam všech dynamických pohledů

  - následujícím příkazem lze zjistit, že aktuální verze 11.2.0 obsahuje celkem **525** V$ pohledů

```
SQL> select count(*) from v$fixed_table
     where name like 'V$%';
```

- V$OBJECT_USAGE – poskytuje informace o využití objektu. Typicky se používá při zjišťování, zda se neplýtvá výkonem na režii spojenou se správou nevyužitého indexu
  - monitorování lze aplikovat na index pomocí následujícího příkazu

```
SQL> alter index nazevindexu monitoring usage;
```

- V$OPTION – poskytuje informace o nainstalovaných dodatečných balíčcích

- V$RECOVER_FILE – poskytuje informace o tom, zda nějaký datový soubor nepotřebuje obnovit
  - následující příkaz zobrazí názvy souborů k obnově a příčiny poškození

```
SQL> select d.name, r.online_status, r.error from
     v$datafile d,v$recover_file r where d.file#=r.file#;
```

- V$SESSION – obsahuje informace o všech právě probíhajících relacích
  - parametry SID a SERIAL# mohou být využity pro násilné ukončení relace příkazem:

```
SQL> alter system kill session 'sid, session-serial';
```

  - následující příkaz zobrazí pořadí aktuální relace v dynamickém pohledu

```
SQL> select sid from v$session
     where audsid = sys_context( 'userenv','SESSIONID' );
```

- V$SGA – zobrazí informace o aktuálním využití paměti SGA

- V$SQL – poskytuje kumulativní statistiky SQL dotazů, obsahuje jeden řádek pro každý nový zaznamenaný SQL příkaz. Statistiky V$SQL jsou aktualizovány po ukončení dotazu, při dlouhotrvajících dotazech jsou aktualizovány každých 5 sekund.
  - následující příkaz vypíše texty sql příkazů ve vyrovnávací paměti seřazené podle náročnosti

```
SQL> select sql_fulltext from v$sql order by cpu_time desc;
```

- V$SQL_MONITOR – zobrazuje SQL dotazy, které jsou monitorovány. SQL monitorování se automaticky aktivuje, když je příkaz zpracováván paralelně či když jeho provádeěcí doba přesáhne 5 sekund. Pokud je monitorovaný dotaz právě prováděn, aktualizace pohledu V$SQL_MONITOR probíhá prakticky v reálném čase.

- V$SYSSTAT – zobrazuje kumulativní statistiky od nastartování databáze. Tento pohled lze např. využít pro výpočet poměru *Hit Ratio*, který udává kolik procent bloků bylo přečteno z harddisku (fyzické čtení) a kolik z paměti.
  - zobrazení poměru *Hit Ratio* lze provést následujícím příkazem

```
SQL> select 100 * (1-(p.value/(b.value+c.value)))
     from v$sysstat p, v$sysstat b, v$sysstat c
```

```
where p.name='physical reads' and b.name='db block gets'
and c.name ='consistent gets';
```

- V$VERSION – zobrazí informace o právě používané verzi Oracle Database
- V$UNDOSTAT – obsahuje informace o využití prostoru pro uložení návratových informací, aktualizované každých 10 minut.

4.1.3 Automatic Workload Repository

Úložiště statistických dat *Automatic Workload Repository* (AWR), poprvé představeno ve verzi 10g je následníkem nástroje STATSPACK. Je určeno pro sběr statistik v předem definovaných časech či intervalech prostřednictvím sad statistických údajů, tzv. *snapshotů*. Na rozdíl od STATSPACKu je vysoce integrováno s nástrojem OEM a lze s ním snadno analyzovat a opravit výkonové komplikace [4].

AWR je uloženo v tabulkovém prostoru SYSAUX, jeho aktivace se provede nastavením inicializačního parametru STATISTICS_LEVEL na hodnotu TYPICAL nebo ALL, možnost BASIC umožňuje pouze manuálně spouštěné shromažďování.

AWR sbírá mimo jiné i tyto statistiky výkonu:

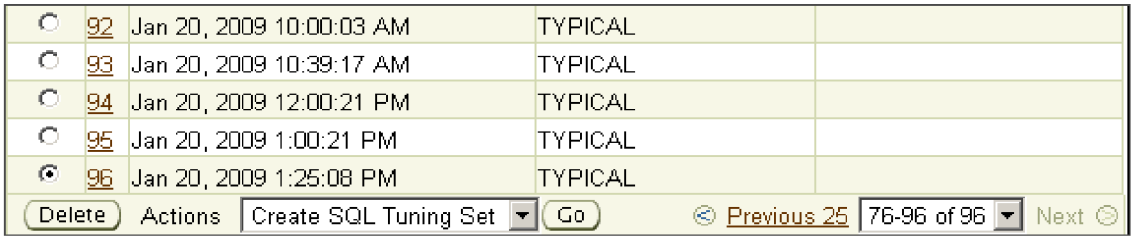
- události čekání (*wait events*) pro analýzu výkonových komplikací
- statistiky časového modelu indikující délku databázového času spojeného s procesem uloženým v dynamických modelech V$SESS_TIME_MODEL a V$SYS_TIME_MODEL
- statistiky aktivních relací (ASH) z dynamického pohledu V$ACTIVE_SESSION_HISTORY
- systémové a relační statistiky z dynamických pohledů V$SYSSTAT a V$SESSTAT
- statistiky využití objektů dynamického pohledu V$OBJECT_USAGE
- statistiky o výkonově náročných SQL příkazech

Statistické údaje jsou implicitně shromažďovány každou hodinu a udržovány v úložišti po dobu sedmi dní. Jednotlivé *snapshoty* uložené v AWR lze prohlížet pohledem datového slovníku *DBA_HIST_SNAPSHOT*.

Manuální vytvoření sady statistických údajů je možné provedením skriptu:

```
SQL>  execute DBMS_WORKLOAD_REPOSITORY.create_snapshot ();
```

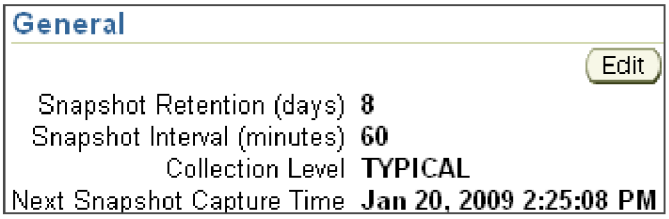
V nástroji OEM Database Control lze tuto funkci najít v sekci *Server* pod odkazem **Automatic Workload Repository** (obr. 4.3).



Obrázek 4.3: Zobrazení *snapshotů* v OEM Database Control

Změna nastavení sběru sad statistických údajů se provede procedurou MODIFY_SNAPSHOT_SETTINGS. Změnit je možné celkovou dobu ukládání informací a interval sběru sad statistických dat. Následujícím skriptem změní interval sběru *snapshotů* na 30 minut a dobu uchovávání dat na 30 dnů (zadáno v minutách):

```
SQL>  execute DBMS_WORKLOAD_REPOSITORY.modify_snapshot_settings
      ( retention => 43200, interval => 30 );
```



Obrázek 4.4: Editace *snapshotů* v OEM Database Control

Jednu či více sad statistických dat v určeném rozmezí lze vymazat následujícím příkazem:

```
SQL> execute DBMS_WORKLOAD_REPOSITORY.drop_snapshot_range
     ( low_snap_id  => 25, high_snap_id => 28 );
```

Určitou množinu statistických údajů je možné využít jako výchozí hodnotu (*baseline*) pro účely porovávání změny výkonu systému v čase. Takto deklarované *snapshoty* budou později k dispozici pro porovnání proti aktuálním sadám dat. Výchozí sadu lze vytvořit následujícím příkazem:

```
SQL>  execute DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE
      ( start_snap_id => 25, end_snap_id => 27,
      baseline_name => 'Streda poledne' );
```

Množina sad statistických údajů asociovaných s vytvořenou výchozí sadou zůstává v úložišti dokud tato není explicitně odstraněna procedurou DROP BASELINE. Parametr CASCADE určuje, zda se s odstraněním baseline automaticky odstraní i odpovídající snapshoty.

```
SQL>  execute DBMS_WORKLOAD_REPOSITORY.DROP_BASELINE
      ( baseline_name => 'Streda poledne', cascade => FALSE );
```

Ve verzi Oracle 11g byly zavedeno několik nových nástrojů pro sledování zátěže pomocí *baseline*. Prvním je sada statistických údajů SYSTEM_MOVING_WINDOW, která umožňuje sledovat zátěž a

výkonnost databázové instance v reálném čase (obr. 3.4). Dalším je nástroj opakujících se *baseline*, které umožňují definovat pravidelný časový rámec, např. pro určitou denní dobu, kdy je plánována velká zátěž. Novou funkcí jsou také adaptivní prahové metriky *Adaptive Metric Thresholds*, které mohou nahradit manuální nastavování metrik jejich automatizováním užitím *baseline* AWR dat.

Kromě výše popsaného dynamického pohledu `V$ACTIVE_SESSION_HISTORY` jsou s úložištěm AWR spojeny následující pohledy [13]:

- `V$METRIC` – zobrazuje informace o metrikách
- `V$METRICNAME` – zobrazuje metriky odpovídající všem skupinám metrik.
- `V$METRIC_HISTORY` – zobrazí historii metrik
- `V$METRICGROUP` – zobrazí všechny skupiny metrik
- `DBA_HIST_ACTIVE_SESS_HISTORY` – zobrazí historii změny obsahu ASH
- `DBA_HIST_BASELINE` – zobrazuje historii výchozích sad *baseline*
- `DBA_HIST_DATABASE_INSTANCE` – zobrazí informace o databázovém prostředí
- `DBA_HIST_SNAPSHOT` – zobrazí informace o sadě statistických údajů
- `DBA_HIST_SQL_PLAN` – zobrazí plány provádění příkazů SQL
- `DBA_HIST_WR_CONTROL` – zobrazí nastavení *Automatic Workload Repository*

Skripty `awrrpt.sql` a `awrrpti.sql` pro manuální generování zpráv na základě uložených snapshotů jsou uloženy v adresáři `$ORACLE_HOME/rdbms/admin`. Zprávy vytvářené těmito skripty mají formát HTML či prostého textu, podobný zprávám nástroje STATSPACK.

4.1.4 Automatic Database Diagnostic Monitoring

Jedním z nástrojů sloužícím k vyhodnocení statistik shromážděným úložištěm AWR je *Automatic Database Diagnostic Monitoring* (ADDM). ADDM je samo-diagnostický engine integrovaný do jádra databázového systému poskytující široké možnosti automatizace řešení výkonových problémů. Představuje přesun úloh administrátora databáze směrem k samo-spravující se databázi. Nástroj ADDM umožňuje administrátovi provést nejen automatickou analýzu databázové výkonnosti a prezentaci symptomů, ale i vygenerování sestavy doporučující, jakým způsobem mohou být identifikované problémy řešeny.

ADDM analyzuje data skladovaná v úložišti AWR a předkládá výsledky ve formě tzv. **nálezů** (*findings*). Typické výkonové komplikace, na které může poukázat nástroj ADDM jsou tyto:

- nastalé konflikty přístupu ke systémovým zdrojům (*bottleneck*), např. v případě, že databáze využívá příliš mnoho výkonu procesoru zpracováváním vysokozátěžových SQL dotazů
- špatné spravování připojování k databázi, typicky v případě nadbytečného opakovaného připojování jedné aplikace k databázové instanci

- konflikty uzamykání dat v prostředí s více uživateli, kdy jeden klient získá datový zámek pro bezpečnou aktualizaci dat, což způsobí čekání dalších procesů

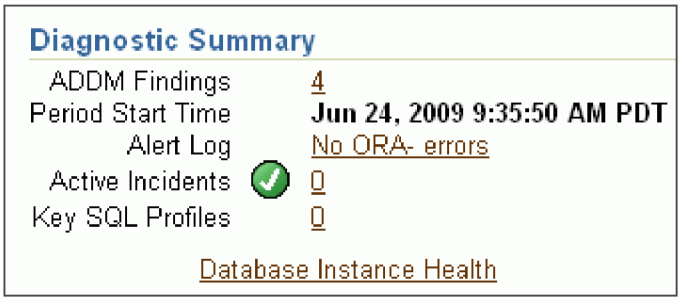
Generování sestav nástroje ADDM je možné spouštět v prostředí OEM Database Control v sekci *Performance* či skriptem `addrmrpt.sql` umístěném ve složce `$ORACLE_HOME/rdbms/admin`. Implicitně je nastaveno automatické generování sestav každou 1 hodinu.

Data získaná analýzou prováděnou nástrojem ADDM lze prohlížet pohledy datového slovníku, graficky reprezentovanými v OEM nástroji Database Control.

Pohledy datového slovníku nástroje ADDM:

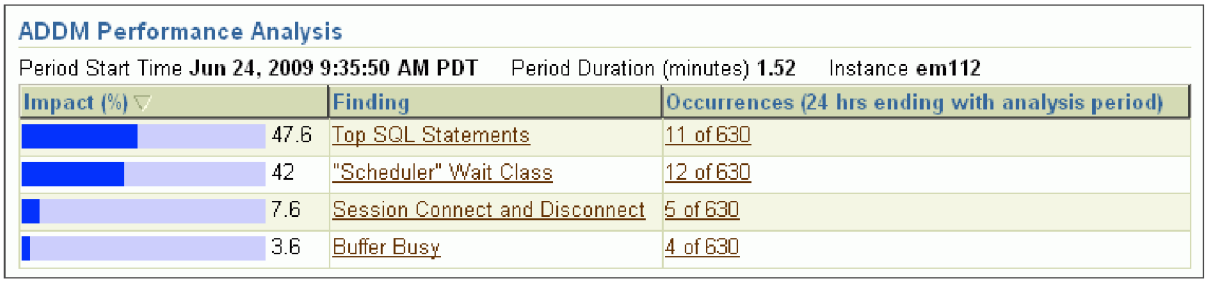
- `DBA_ADVISOR_TASK` – obsahuje existující úlohy
- `DBA_ADVISOR_LOG` – obsahuje stav a průběh úloh
- `DBA_ADVISOR_RECOMMENDATIONS` – obsahuje dokončené diagnostické úlohy a doporučení
- `DBA_ADVISOR_FINDINGS` – obsahuje informace o nálezech

Diagnostické shrnutí lze nalézt na úvodní stránce OEM Database Control (obr. 4.5).



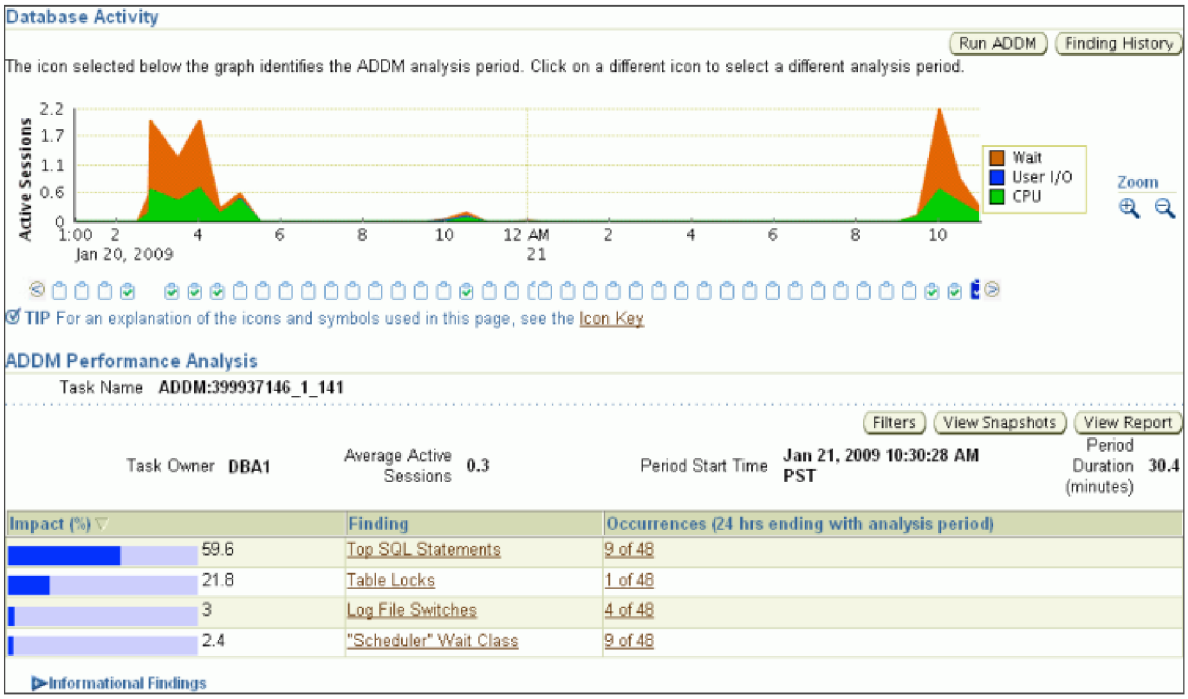
Obrázek 4.5: Shrnutí nálezů ADDM

Shrnutí výkonové analýzy je umístěno na hlavní stránce pod sekcí *Alerts* (obr. 4.6).



Obrázek 4.6: Shrnutí výkonové analýzy ADDM

Po kliknutí na číselný odkaz vedle *ADDM Findings* se zobrazí detailní stránka výsledků běhu ADDM (obr. 4.6)



Obrázek 4.7: Shrnutí nálezů ADDM

Stránka ADDM ukazuje aktivitu databáze během zadaného intervalu výkonové analýzy. Typy databázových aktivit jsou definovány v legendě vpravo od grafu výkonu. Ikony pod grafem reprezentují každá jednu ADDM úlohu odpovídající dvojici *snapshotů* skladovaných v úložišti AWR.

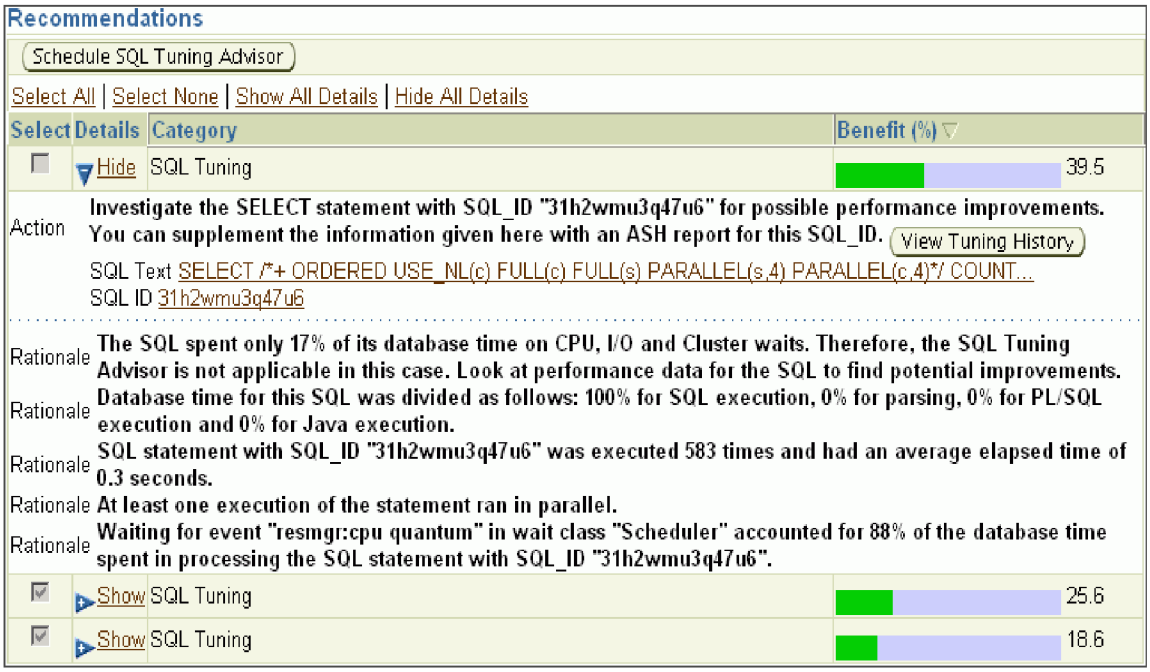
V příkladu na obr. 4.7 vidíme dva bloky databázové aktivity mezi 2:30 a 5:30, respektive od 9:30. Převaha událostí čekání (*Wait*) indikuje, že na výkon databáze měly dominantní vliv události čekání. Naopak v časovém úseku od 4:30 je patrné, že na výkon databáze mohl mít negativní vliv nedostatečný výkon CPU.

Nálezy nástroje ADDM mohu být třech typů:

- *Problém* – nález popisuje příčinu výkonového problém databázemi
- *Symptom* – nález obsahuje informace vedoucí k nálezům problémů
- *Informace* – nález zpravuje administrátora o stavech částí systému, které nemají negativní dopady na výkonnost

Každý nález problému je kvantifikován odhadovanou částí databázového času, která byla důsledkem výkonové komplikace. Pokud se příčiny problémového nálezu překrývají, může celkový součet jejich procentuálních dopadů překročit hranici 100%. Například v případě vysokozátěžového SQL dotazu může být 50% databázového času způsobeno I/O diskovými operacemi a 75% nedostatečnou velikostí vyrovnávací paměti.

U nálezů způsobujících potíže s výkonem navrhne ADDM řešení aplikovatelné nástroji pro ladění výkonnosti popsané v kapitole 4.2 (obr 4.8).



Obrázek 4.8: Doporučení nástroje ADDM

4.2 Nástroje pro ladění výkonnosti systému

Cílem návrhu databáze je, aby fyzická omezení aplikací a příslušný hardware (propustnost I/O diskových operací, velikost paměti, výkon dotazů) neměly vliv na provádění funkcí aplikace [1]. Výkonová omezení vyplývající z limitu aplikačního prostředí (použitý hardware, síťová spojení) je třeba zohlednit již při návrhu aplikace. Nároky na výkonnost databázového systému by ideálně měly být pokryty s výraznou rezervou, čehož lze dosáhnout jeho optimalizací, tedy **laděním**.

Pro efektivní ladění databázového systému je třeba věnovat se následujícím oblastem, ideálně v uvedeném pořadí [4]:

- Vyhodnocení návrhu aplikace
- Ladění příkazů jazyka SQL
- Ladění využití paměti
- Ladění využití úložiště dat
- Ladění příkazů pro manipulaci s daty
- Ladění fyzického a logického úložiště
- Ladění síťového provozu

Databázový systém Oracle 11g poskytuje pro ladění automatizované nástroje - poradce (*Advisors*) integrované v rozhraní OEM Database Control. Jsou dostupné v sekci *Related Links* pod odkazem **Advisor Central**.

Poradci pro ladění SQL		(*SQL Advisors*)
SQL Access Advisor	•	Vyhodnotí celkovou SQL zátěž a navrhne vytvoření indexů, materializovaných pohledů či partitioning objektů zlepšující celkovou výkonnost SQL zátěže nad schematem databáze
SQL Tuning Advisor	•	Analyzuje jednotlivé příkazy jazyka SQL nebo jejich soubor (*SQL Tuning Set*) a navrhuje použití SQL profilů, indexů a dalších doporučení pro zlepšení výkonnosti
	•	Je automatizován pro spuštění během periodické údržby databáze
Poradci pro správu paměti		(*Memory Advisors*)
Memory Advisor SGA Advisor Shared Pool Advisor Buffer Cache Advisor PGA Advisor	•	Poradci pro správu paměti poskytují grafickou analýzu nastavení parametrů paměťových oblastí SGA a PGA. Jejich použití je podmíněno aktivací možnosti automatické správy paměti *Automatic Memory Management*
Další poradci		
Segment Advisor	•	Analyzuje objekt databáze a určuje segmenty vhodné k defragmentaci
Undo Advisor	•	Poskytuje ladicí informace pro návratové tabulkové prostory a pro optimální nastavení parametru UNDO_RETENTION

Tabulka 4.1: Přehled nástrojů *Advisors* Oracle 11g

Tato kapitola se věnuje možnostem, které poskytuje Oracle 11g pro ladění příkazů jazyka SQL. Konfigurační nástroje pro správu paměti, diskového prostoru a manipulaci s daty jsou popsány v kapitole 4.3

4.2.1 Úvod do ladění příkazů jazyka SQL

Návrh aplikace a problémy s příkazy jazyka SQL způsobují většinu problémů s výkonem v jinak správně navržené a nakonfigurované databázi [4]. Je tedy vhodné, aby se kromě ladicích úkonů podílel administrátor databáze i na návrhu SQL příkazů, které aplikace využívá.

Cílem ladění SQL je minimalizace cest, které aplikace využívá pro nalezení identifikátoru *RowID*, který obsahuje informace o konkrétním fyzickém uložení hledaného řádku tabulky (datový soubor, blok v rámci souboru a řádek v rámci bloku). Je-li nad poměrně velkou tabulkou prováděn SQL příkaz s klauzulí **where**, pravděpodobně lze ladit výkon databáze definováním *indexu* nad jedním či více sloupci tabulky.

Index mapuje logické hodnoty tabulky na odpovídající identifikátory *RowID*. Optimalizátor vyhledávacích cest pak může využít např. přístupu úplného průchodu indexem, funkci *skip-scan* (neprovádění úplného průchodu tabulkou) či optimalizaci *index range scan* (dotazování hodnot v určitém rozsahu). Je důležité, aby data tabulky byla v co největší míře seřazená, což v praxi znamená, že je při provádění dotazu nutné fyzicky číst jen malý počet bloků [4]. Naopak čím více indexů je nad tabulkou vytvořeno, tím déle se budou provádět všechny operace insert, update a delete.

Vícesloupcový index nad tabulkou zaměstnanců lze vytvořit např. tímto příkazem:

```
SQL>   create index emp_01_ind
       on EMPLOYEES ( empid, deptid ) tablespace INDEXES;
```

Pokud je rozmanitost uložených dat malá, je možné použít *rastrové indexy*, které slouží k efektivnímu mapování několika málo hodnot. Další možností je vytvoení *clusteru*, který ukládá řádky z různých tabulek v závislosti na jejich logických hodnotách (klíč clusteru) do stejných datových bloků. Při častých dotazech na rovnost hodnot je výhodné použití *hašovaného clusteru*, který ukládá řádky tabulky do fyzických umístění podle hodnoty klíče.

Hašovaný cluster s klíčem ve sloupci `empid` lze vytvořit následujícím příkazem:

```
SQL>   create cluster emp_cluster ( empid NUMBER(5,0) )
       PCTUSED 80 PCTFREE 5 TABLESPACE users
       STORAGE ( INITIAL 250K NEXT 50K MINEXTENTS 1 MAXEXTENTS 3
       PCTINCREASE 0 ) HASH IS empid HASHKEYS 150;
```

A vytvoření tabulky:

```
SQL>   create table EMPLOYEES ( empid NUMBER(5,0) PRIMARY KEY,
       ... ) CLUSTER emp_cluster (empid);
```

Dalšími typy indexů jsou např. *indexy reverzní* (uložení dat na základě reverze jejich hodnot), *indexy založené na funkcích* či *textové indexy* (spravování seznamů a výskytů slov).

4.2.2 Prováděcí plány příkazů SQL

Prováděcí plán (*Query Execution Plan*) je generován databází Oracle pro každý zadaný příkaz SQL a obsahuje informace o způsobu zpracování dotazu. Tyto informace lze zjitit příkazem **explain plan**, případně nastavit automatické zobrazování prováděcích informací bez spuštění dotazu příkazem **set autotrace on**. Výsledky se uloží do tabulky PLAN_TABLE, kterou lze manuálně vytvořit pomocí skriptu `utlxplan.sql`, uloženého ve složce `$ORACLE_HOME/rdbms/admin`.

Prováděcí plán pro SQL příkaz lze generovat např. tímto skriptem:

```
SQL> explain plan for
     select … from PRODUCTS p, INVENTORIES i, WAREHOUSES w
     where p.pid = i.pid and i.wid = w.id
     group by warehouse_name, product_name order by quantity;
```

Dotaz nad plánovací tabulkou lze provést pomocí procedury DBMS_XPLAN:

```
SQL>   select * from table( DBMS_XPLAN.DISPLAY );
```

V nástroji OEM Database Control vypadá prováděcí plán pro výše uvedený příkaz takto:

Operation	Line ID	Object	Object Type	Order	Rows	Bytes	Cost	Time	CPU Cost	I/O Cost
SELECT STATEMENT	0			16		92,305	102	2	154,058,448	97
SORT ORDER BY	1			15		92,305	102	2	154,058,448	97
HASH GROUP BY	2			14		92,305	102	2	154,058,448	97
FILTER	3			13						
HASH JOIN RIGHT OUTER	4			12		92,305	100	2	88,339,184	97
VIEW	5	OE.index$_join$_005	VIEW	4		11,250	88	2	22,411,900	87
HASH JOIN	6			3						
INDEX FAST FULL SCAN	7	OE.PRD_DESC_PK	INDEX (UNIQUE)	1		11,250	41	1	2,453,759	41
INDEX FAST FULL SCAN	8	OE.PROD_NAME_IX	INDEX	2		11,250	68	1	2,640,697	68
HASH JOIN	9			11		48,867	12	1	49,596,548	10
TABLE ACCESS FULL	10	OE.PRODUCT_INFORMATION	TABLE	5		5,625	5	1	141,539	5
MERGE JOIN	11			10		27,148	6	1	33,124,274	5
TABLE ACCESS BY INDEX ROWID	12	OE.WAREHOUSES	TABLE	7		0,132	2	1	17,753	2
INDEX FULL SCAN	13	OE.WAREHOUSES_PK	INDEX (UNIQUE)	6			1	1	8,921	1
SORT JOIN	14			9		10,859	4	1	33,106,520	3
TABLE ACCESS FULL	15	OE.INVENTORIES	TABLE	8		10,859	3	1	246,887	3

Obrázek 4.9: Prováděcí plán SQL příkazu

Pořadí přístupů k indexům, tabulkám či pohledům je uveden ve sloupci *Order*. Každá operace obsahuje ve sloupcích *Cost* informace o její náročnosti. Tyto údaje je možné využít při hledání kroků, které zaberou největší část celkové náročnosti dotazu a tyto kroky dále ladit [4].

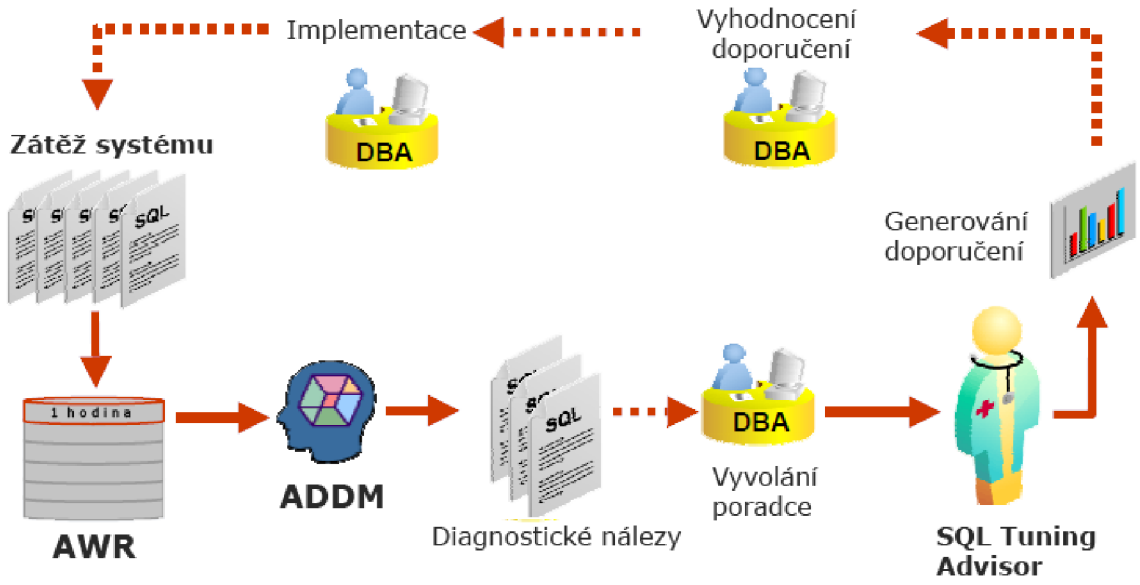
4.2.3 SQL Tuning Advisor

Poradce pro ladění příkazů SQL (*SQL Tuning Advisor*) je od verze Oracle 10g součástí dodatečného balíčku *Oracle Tuning Pack* [14]. Slouží k automatizaci procesu ladění SQL zevrubný zkoumáním všech možných způsobů ladění příkazu jazyka SQL. Typicky se provádí na nálezech doporučených k řešení nástrojem ADDM (viz. kapitola 4.1.4). Analýza a ladění je ve verzi 11g prováděna výrazně vylepšeným optimalizátorem dotazů. *SQL Tuning Advisor* provádí 4 základní typy analýzy [15]:

- **Analýza statistik** – identifikuje objekty se zastaralými či chybějícími statistikami a doporučí řešení aktualizace
- **SQL profilování** – navrhuje vytváření a užití *profilů SQL*. SQL profily jsou kolekce statistik umožňující optimalizátoru SQL dotazů vytvořit ideální prováděcí plán. Eliminuje potřebu manuálních řešení a náročných změn v aplikačním kódu pro optimalizaci příkazů jazyka SQL.
- **Analýza přístupových cest** – navrhuje vytvoření nových indexů zlepšujících výkon provádění SQL dotazů
- **Analýza struktury SQL** – navrhuje zlepšení výkonu restrukturalizací příslušného příkazu jazyka SQL

Ve verzi 11g databáze Oracle byl nástroj *SQL Tuning Advisor* vylepšen o srovnávání pozitivních dopadů jednotlivých SQL profilů na výkon. Při zjištění zastarelého SQL profilu neodpovídajícímu současnému stavu systému (např. po upgrade) aplikuje na provádění SQL dotazu více SQL profilů a vyhodnotí jejich srovnání.

Životní cyklus ladění příkazů jazyka SQL je zobrazen na obr. 4.10 [14].



Obrázek 4.10: Životní cyklus ladění příkazů jazyka SQL (*Převzato z* [14])

Nástroj SQL Tuning Advisor lze spouštět oproti následujícím zdrojům dat:

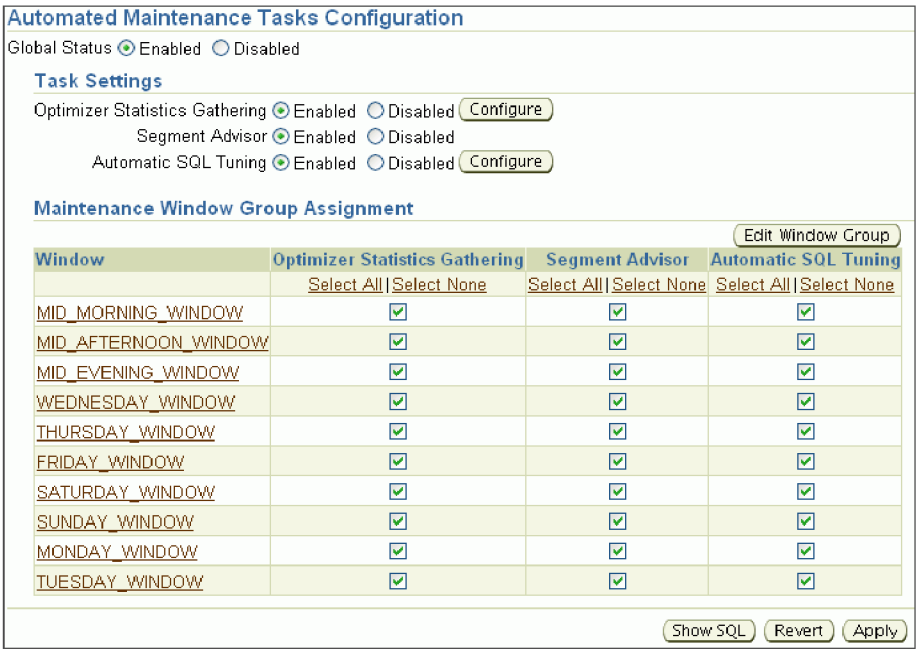
- **Top Activity** – seznam SQL příkazů s největšími nároky na systémové prostředky během poslední hodiny. Lze jej zobrazit v OEM Database Control ze sekce *Performance*.
- **Historical SQL** – sada SQL příkazů během vybraného 24hodinového intervalu. Vhodná k proaktivnímu ladění databáze
- **SQL Tuning Sets** – vlastní sada SQL příkazů

4.2.4 Automatizace ladění příkazů SQL

Ve verzi databáze Oracle 11g byla v rámci snahy o zvýšeni samo-spravovatelnosti databáze nově implementována možnost automatizovat spouštění nástroje *SQL Tuning Advisor* během časových intervalů oken údržby systému (MAINTENANCE_WINDOW_GROUP) [14]. Poradce během každého spuštění identifikuje SQL dotazy generující největší zátěž systému a generuje doporučení, jak tyto dotazy ladit.

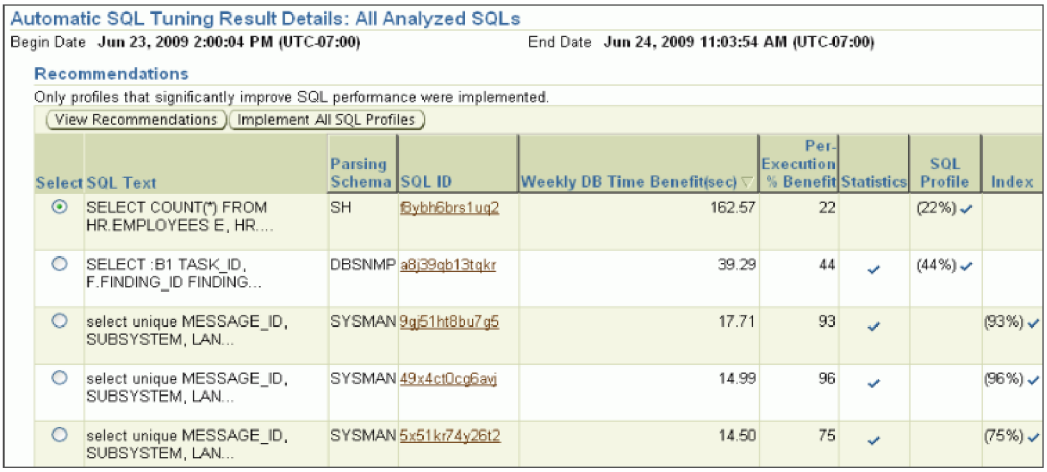
Nástroj automatického ladění SQL vybírá z úložiště AWR shromážděné nálezy ve formě kandidátních příkazů SQL na základě jejich dopadu v různých časových intervalech (týdenní, denní, hodinové...). Na seznam těchto kandidátních příkazů pak aplikuje vybraný SQL profil a testuje jej způsobem uvedeným v kapitole 4.2.3. Nástrojem OEM Database Control lze nastavit automatické aplikování SQL profilů, které mají na provádění příkazů dostatečný pozitivní dopad (3X v času CPU a I/O [14]). V případě, že jejich benefit je menší, pouze jej doporučí ke zvážení administrátorovi databáze.

Automatické spouštění ladění příkazů SQL lze spustit v sekci *Oracle Scheduler*, odkazem **Automatic Maintenance Task** (obr. 4.11).



Obrázek 4.11: Automatizace údržby systému

Seznam laděných příkazů a implementovaných či doporučených profilů lze zobrazit odkazem **View Report** v sekci *Automatic SQL Tuning Result Summary*.



Obrázek 4.12: Výsledky spuštění nástroje Automatic SQL Tuning

Automatickou implementaci SQL profilu značí zelená značka ve sloupci *SQL Profile*.

4.2.5 Monitorování SQL v reálném čase

Ve verzi databáze Oracle 11g je nově implementován nástroj pro sledování SQL příkazů s vysokou prodlevou odezvy v reálném čase. Tento nástroj automaticky monitoruje příkazy, pokud jejich

provádění překročí 5 sekund času CPU nebo I/O operací či jsou-li prováděny paralelně. Umožňuje tedy sledovat výsledky ladění SQL s minimální prodlevou po implementaci ladicích opatření. Každý příkaz jazyka SQL je monitorován samostatně [14].

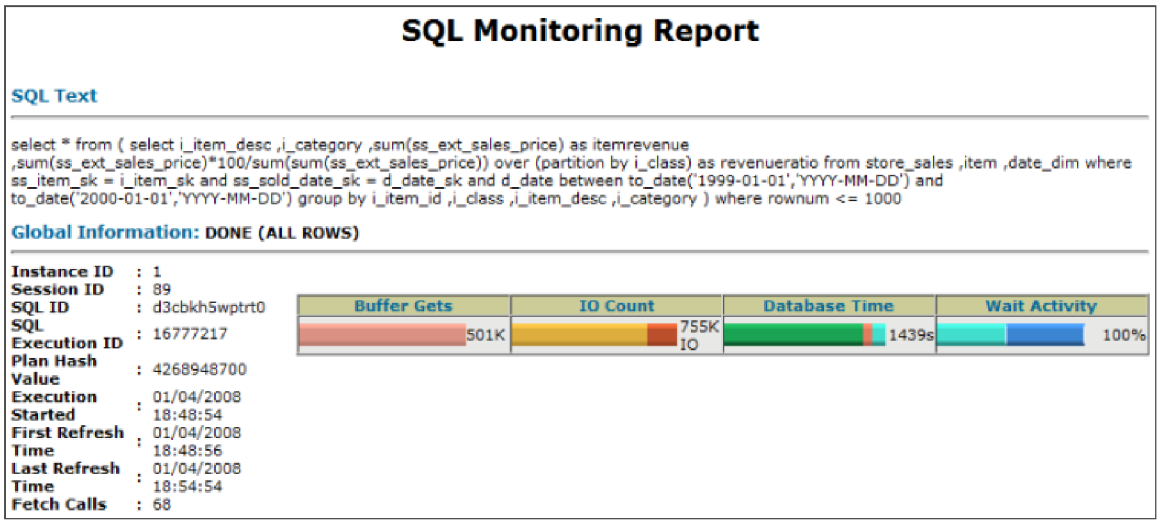
Statistiky sledování jsou k dispozici prostřednictvím těchto dynamických pohledů:

- `V$SQL_MONITOR` – poskytuje kumulativní statistiky využití databázového času rozdělené do skupin (CPU, I/O, *Application*, …)
- `V$SQL_PLAN_MONITOR` – zobrazuje statistiky operací prováděcího plánu SQL dotazu
- `V$ACTIVE_SESION_HISTORY` – každý příkaz SQL je identifikovatelný pohledem ASH (viz. kapitola 4.1.2)

Veškeré statistiky jsou obnovované v intervalu 1 sekundy a jsou přístupné minimálně po dobu 5 minut. Zprávy monitorování v reálném čase je možné získat procedurou `DBMS_SQLTUNE.REPORT_SQL_MONITOR`:

```
SQL>  select dbms_sqltune.report_sql_monitor from dual;
```

Grafickou reprezentaci zprávy je možné získat nástrojem OEM Database Control v sekci *Additional Monitoring Links* na obrazovce *Performance* pod odkazem **SQL Monitoring** (obr. 4.13).



Obrázek 4.13: Hlavička zprávy o monitorování SQL v reálném čase

4.3 Nástroje pro správu systému

Tato kapitola se věnuje nástrojům, které poskytuje databázový systém Oracle 11g pro účely správy využitelného místa a systémových prostředků. Informace o nástrojích prezentovaných v této kapitole byly vyjma oficiální dokumentace Oracle 11g [11] čerpány z publikace [4].

4.3.1 Správa diskového místa

Problémy se správou místa se typicky dají shrnout do 3 kategorií: nedostatek místa pro tabulkové prostory, nedostatek místa pro návratové objekty u dlouhotrvajících dotazů, nedostatek místa pro dočasné segmenty a fragmentace segmentů či tabulkových prostorů.

Tabulkový prostor je omezen součtem velikostí datových souborů, ve kterých je uložen. Použitím atributu tabulkového prostoru AUTOEXTEND, nastavením správy rozšíření na hodnotu LOCAL, alokace prostoru na AUTOALLOCATE a správy segmentového prostoru na AUTO lze zajistit, že při vždy v případě zaplnění datových souborů dojde k jejich rozšíření skokově o zadanou hodnotu (implicitně 100MB). Úlohou administrátora databáze je sledovat trendy využití místa v čase, aby bylo možné pokrýt jakékoli budoucí požadavky na velikost volného místa.

Dočasné segmenty slouží k ukládání mezivýsledků databázových operací, které nelze provést v paměti, např. operace řazení, klauzulí **distinct** či sjednocení **union**. Všem uživatelům databáze je třeba přiřadit tabulkový prostor pro ukládání dočasných segmentů, případně jej definovat implicitně (typicky TEMP). Operace, kterým nejsou během vykonávání z důvodu nedostatku místa v dočasném tabulkovém prostoru dostupný dočasné segmenty, selžou.

Podobně je třeba věnovat pozornost velikosti *návratových tabulkových prostorů*. Slouží k vrácení změn nepotvrzených databázových transakcí a zajišťují konzistentní obraz čtení dat pro dlouhotrvající dotazů.

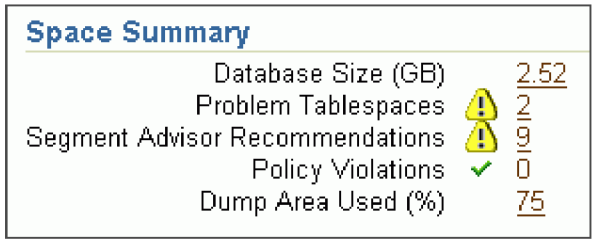
Sledování využití diskového prostoru zajišťují zejména pohledy datového slovníku, podrobněji představené v kapitole 4.1.1. Jsou to zejména `DBA_TABLESPACES`, `DBA_SEGMENTS`, `DBA_EXTENTS`, `DBA_FREE_SPACE`, `DBA_LMT_FREE_SPACE` a `DBA_THRESHOLDS`.

Ve výchozí instalaci systému Oracle Database 11g jsou všechny tabulkové systémy vytvářeny jako *místně řízené*. Tento typ tabulkového prostoru spravuje volné místo pomocí *rastrových map*, které do značné míry eliminují problém fragmentace. Přesto mohou v důsledku provádění operací **insert** a **update** jednotlivé segmenty tabulek a indexů obsahovat velké množství volného místa.

Diskové místo lze účinně uvolnit defragmentačním nástrojem **Segment Advisor**. Je toho dosaženo následujícími dvěma operacemi:

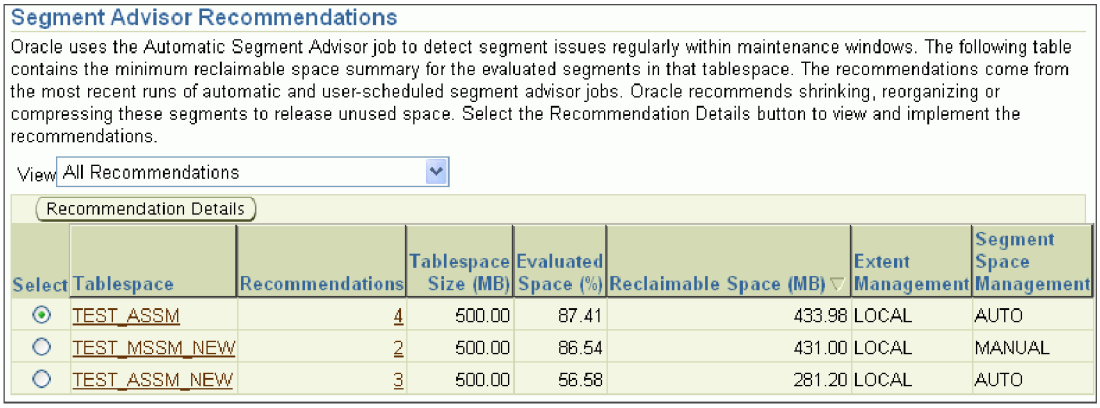
- **Defragmentace segmentu** – operuje na segmentech tabulky a objektů na ní závislých (indexy, rozdělení...). Data obsažená v segmentu jsou sdružena na jeho začátek a uvolněné místo je vráceno tabulkovému prostoru nebo je ponecháno segmentu pro budoucí operace vkládání. Defragmentaci nelze provádět v slovníkově řízených tabulkových prostorech a tabulkových prostorech s aktivovanou manuální správu segmentového prostoru.
- **Reorganizace tabulky** – přesune tabulku a objekty na ní závislé na jiné místo tabulkového prostoru. Na rozdíl od defragmentace je tedy nutnou prerekvizitou této operace dostatečné volné místo minimálně o velikosti reorganizované tabulky.

Shrnutí informací o diskovém místě lze nalézt na hlavní stránce OEM Database Control (obr. 4.14).



Obrázek 4.14: Shrnutí informací o správě místa

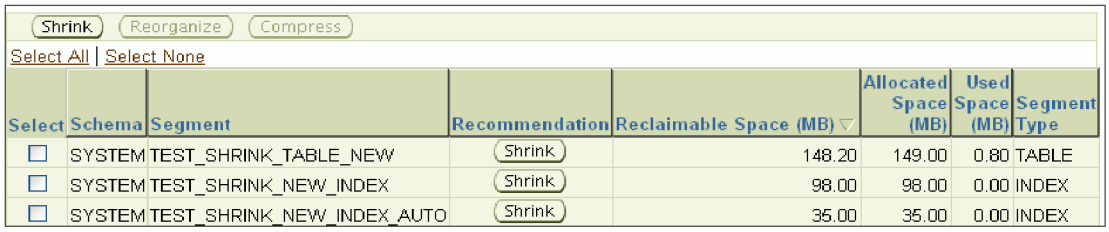
Doporučení nástroje lze zobrazit kliknutím na odkaz *Segment Advisor Recommendation* (obr 4.15)



Select	Tablespace	Recommendations	Tablespace Size (MB)	Evaluated Space (%)	Reclaimable Space (MB)	Extent Management	Segment Space Management
◉	TEST_ASSM	4	500.00	87.41	433.98	LOCAL	AUTO
○	TEST_MSSM_NEW	2	500.00	86.54	431.00	LOCAL	MANUAL
○	TEST_ASSM_NEW	3	500.00	56.58	281.20	LOCAL	AUTO

Obrázek 4.15: Doporučení nástroje Segment Advisor

Jednotlivé defragmentační operace zobrazí kliknutí na *Recommendation Details* (obr. 4.16)



Select	Schema	Segment	Recommendation	Reclaimable Space (MB)	Allocated Space (MB)	Used Space (MB)	Segment Type
☐	SYSTEM	TEST_SHRINK_TABLE_NEW	Shrink	148.20	149.00	0.80	TABLE
☐	SYSTEM	TEST_SHRINK_NEW_INDEX	Shrink	98.00	98.00	0.00	INDEX
☐	SYSTEM	TEST_SHRINK_NEW_INDEX_AUTO	Shrink	35.00	35.00	0.00	INDEX

Obrázek 4.16: Doporučené defragmentační operace

Použití nástroje **Undo Advisor** pro správu návratových tabulkových prostorů je podobné s tím rozdílem že je nutné určit časový rozsah provedení analýzy zadáním identifikátorů *snapshotů* z úložiště AWR (viz. kapitola 4.1.3). Spustit nástroj *Undo Advisor* lze odkazem **Automatic Undo Management** v sekci *Advisor Central*.

Pro problémy spojené s nedostatkem volného místa pro déletrvající operace poskytuje dále databáze Oracle od verze 9i technologii **Resumable Space Allocation**, která umožňuje pozastavit provádění příkazu dokud není problém vyřešen.

4.3.2 Správa paměti

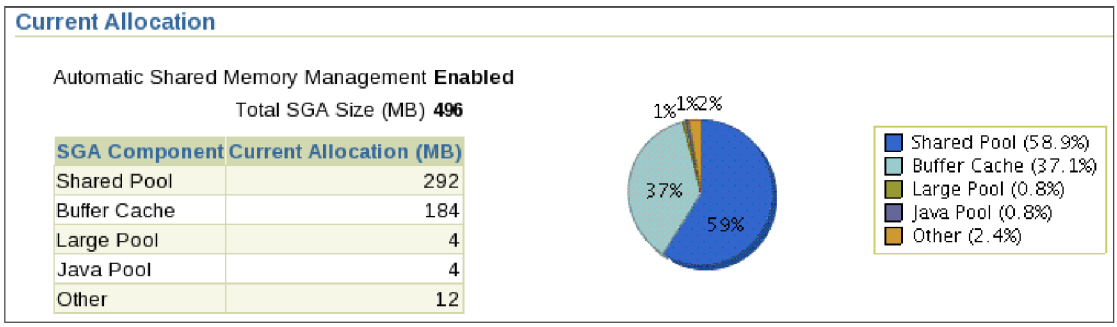
Počínaje verzí 11g má administrátor databáze možnost přenechat správu paměti plně nástroji **automatické správy paměti**. Jediným parametrem, který je při aktivaci tohoto nástroje třeba nastavit je celková velikost paměti přiřazená databázové instanci. Databáze Oracle automaticky nastaví velikost paměťových oblastí SGA (*System Global Area*) včetně jejích komponent a PGA (*Program Global Area*).

Při pokročilé instalaci administrátor vybírá z jednoho z možných módů správy paměti (tab. 4.1).

Mód správy paměti	**Nastavení administrátorem**	**Automatické nastavení**
Automatická správa paměti	• Celková velikost paměti alokované této instanci • Maximální velikost paměti pro tuto instanci (volitelné)	• Celková velikost SGA • Velikost komponent SGA • Velikost PGA této instance • Velikosti individuálních PGA
Automatická správa sdílené paměti a automatická správa paměti SGA	• Cílová velikost SGA • Maximální velikost SGA (volitelné) • Cílová velikost PGA této instance	• Velikost komponent SGA • Velikost individuálních PGA
Manuální správa sdílené paměti a automatická správa paměti PGA	• Velikost sdílené oblasti • Velikost vyrovnávací paměti • Velikost oblasti paměti Java • Velikost oblasti paměti Large • Cílová velikost PGA instance	• Velikost individuálních PGA

Tabulka 4.1: Módy správy paměti

Pokud administrátor databáze obdržel poplach, doporučení nástroje ADDM či potřebuje preventivně nastavit vyšší hodnotu alokované paměti, může využít jednoduchých nástrojů **Memory Advisors** v sekci *Advisor Central*. Aktivace manuální správy paměti PGA je možné, ale **nedoporučuje se** a nelze provést v prostředí OEM Database Control. Na obr. 4.17 lze vidět ukázku grafu hodnot velikosti paměti alokované jednotlivým komponentám paměti SGA.



Obrázek 4.17: Paměť alokovaná komponentám SGA

5 Návrh aplikace

5.1 Úvod do projektu

Cílem ukázkové aplikace *SQLTuningTool* je prezentace proaktivního monitorování databázové SQL zátěže serveru Oracle 11g a nástrojů pro automatické ladění příkazů SQL uvedených v kapitole 4.

Tyto úkony vyžadují specifické oprávnění přístupné pouze administrátorům databáze. Z pochopitelných důvodů nebylo možno pro účely tohoto projektu získat administrátorská oprávnění k databázi s reálným využitím. Pro účely simulace zátěže je postačující provádět dotazování nad *Sample Schemas* Oracle, konkrétně schematem OE.

Základním požadavkem na klientské prostředí je, aby mohlo být dostatečně flexibilní a co nejlépe odpovídalo zátěži reálných systémů. Důležitými parametry takto uměle vyvolávané zátěže jsou počet klientů připojujících se paralelně a nezávisle na sobě k databázi, časové parametry jejich aktivity a definice jimi používaných příkazů jazyka SQL. Pro účely monitorování je důležité, aby byl uživatel schopen měnit tyto parametry intuitivně a v reálném čase. To klade požadavky na snadnou a rychlou ovladatelnost aplikace, je tedy třeba ji vybavit možností zadávání těchto parametrů prostřednictvím vhodného uživatelského rozhraní.

Vzhledem k tomu, že v reálných systémech může být zátěž rozložena velice nerovnoměrně, je žádoucí implementace algoritmů, které budou SQL dotazy generovat náhodně. Uživatelské možnosti rozšíří i možnost jednoduše zadávat a editovat vlastní příkazy a zadávat jejich spouštění v určeném poměru.

K monitorování a ladění systému je na rozdíl od pouhého dotazování databáze potřeba mnohem větší oprávnění. Je tedy potřeba důsledně rozdělit klientské a administrátorské prostředí ukázkové aplikace a umožňovat připojení k databázi klientskými účty s různými oprávněními.

Ukázková aplikace tedy musí bez problémů zvládat režii velkého množství nezávisle běžících paralelních procesů určených k různým účelům. V ideálním případě by simulace zátěžového prostředí a jeho monitorování pravděpodobně neměla sdílet stejné systémové prostředky, pro účel tohoto projektu však byl zvolen formát jedné integrované aplikace.

5.2 Simulace SQL zátěže

Pro modelování zátěže je třeba si uvědomit, které uživatelské faktory ovlivňují výkonnost databáze. Každá klientská relace začíná připojením k databázi, s čímž souvisí režie pro vytvoření aktivní relace.

Tato relace pak může nabývat nstavů WAITING (čekání na zpřístupnění systémových prostředků) či ON CPU.

Základním parametrem pro modelaci klientského prostředí je **počet procesů** připojujících se k databázi. Aplikace spravuje vytváření, spouštění a rušení paralelně běžících vláken, každé obsahující individuální parametry klienta.

Vzhledem k tomu, že účelem ukázkové aplikace je monitorování a ladění příkazů jazyka SQL není důležité vyvolávat zátěž často opakovaným připojováním k databázi. Proto je každému klientovi přiřazen parametr počtu **životních cyklů**, který určuje kolikrát se klient před svým odpojením dotáže databáze. I pro tento parametr je kvůli dosažení co největší reálnosti vhodné implementovat algoritmus náhodného generování.

Dalším parametrem simulačního programu je **interval příchodu** nových klientů do systému. Je vhodné jej implementovat jako uživatelský vstup. Jeho úprava slouží ke změně charakteru klientského prostředí od intenzivní zátěže s připojeným maximálním počtem klientů po rovnoměrně rozloženou zátěž s nestálým počtem aktivních relací.

Volitelným parametrem je **interval mezi** jednotlivými **příkazy** klienta. Všechna klientská vlákna sdílejí stejné paměťové prostředky systému, proto je optimálním řešením ponechat i toto nastavení na uživateli, který si jej určí dle poměru výkonu svého systému a počtu spuštěných klientů.

Samotné **příkazy jazyka SQL** je nutné pro věrnou simulaci generovat náhodným či pseudonáhodným způsobem. K tomu je potřeba vyextrahovat z databáze metadata o dotazovaném schematu. Doplňkem k automatickému generování může být předdefinování vytvořených SQL příkazů a implementace jednoduché editace příkazů vlastních.

Simulace SQL zátěže databáze se tedy skládá z *n* aktivních či neaktivních klientů, připojujících se v určitém intervalu, se zadanými následujícími parametry:

- maximální počet provedených SQL příkazů
- interval mezi prováděním příkazů
- poměr využívání náhodně generovaných a předdefinovaných příkazů

5.3 Monitorování zátěže

Pro sledování výkonu je nutné vytvořit samostatné vlákno konstantně připojené k databázi s administrátorským oprávněním, které v periodických intervalech sbírá aktuální data.

K monitorování náročnosti jednotlivých SQL příkazů je možné využít dynamický pohled `V$SQL`, který je obnovován v reálném čase okamžitě po ukončení dotazu. Tento pohled obsahuje kumulativní statistiky pro každý provedený dotaz. Přehled vybraných statistik poskytuje tabulka 5.1 [11].

Sloupec	Popis
SQL_TEXT	Prvních 1000 znaků příkazu
SQL_FULLTEXT	Plný text SQL příkazu jako sloupec datového typu CLOB
SQL_ID	Jednoznačný identifikátor SQL ve vyrovnávací paměti
SHARABLE_MEM	Počet bajtů sdílené paměti využitých synovským kurzorem
SORTS	Počet řazení provedených pro synovský kurzor
FETCHES	Celkový počet operací získání dat spojený s jedním příkazem
EXECUTIONS	Počet provedení příkazu od odby, kdy byl uložen ve vyrovnávací paměti
LOADS	Údaj, kolikrát byl objekt načten či obnoven
PARSE_CALLS	Počet volání parsování
DISK_READS	Počet provedených fyzických čtení
BUFFER_GETS	Počet provedených čtení z vyrovnávací paměti
USER_IO_WAIT_TIME	Čas čekání z důvodu I/O operací
ROWS_PROCESSED	Počet řádků, které vrátil parsovaný SQL příkaz
COMMAND_TYPE	Definice typu příkazu
HASH_VALUE	Hodnota hashe SQL příkazu ve vyrovnávací paměti
SERVICE	Název služby
CPU_TIME	Čas v mikrosekundách stráveným CPU parsováním, prováděním a fetch
ELAPSED_TIME	Celkový čas strávený parsováním, prováděním a získáváním dat
OBJECT_STATUS	Status kurzoru
LAST_LOAD_TIME	Čas, kdy byl kurzor nahrán do vyrovnávací paměti
SQL_PROFILE	SQL profil
LAST_ACTIVE_TIME	Čas, kdy byl prováděcí plán naposled aktivní

Tabulka 5.1: Statistiky příkazu SQL pohledu V$SQL

Pro účely aplikace není třeba využít všech uložených statistik, z demonstračního pohledu jsou asi nejzajímavějšími parametry SQL_TEXT, SQL_ID, FETCHES, EXECUTIONS, DISK_READS, BUFFER_GETS, CPU_TIME, ROWS_PROCESSED a LAST_ACTIVE_TIME pro filtrování nejaktuálnějších dotazů.

5.4 Ladění SQL příkazů

Rozhraní pro ladění SQL příkazů představuje balíček DBMS_SQLTUNE. Poskytuje aplikaci funkce propojené s množstvím nástrojů uvedených v kapitole 4.2. Jeho použití má 4 fáze:

- **Vytvoření ladicí úlohy** - CREATE_TUNING_TASK

  - vzhledem k tomu, že dynamický pohled V$SQL obsahuje identifikátor příkazu SQL_ID, jako vhodné se jeví využití následující formy tohoto subprogramu:

```
DBMS_SQLTUNE.CREATE_TUNING_TASK (
     sql_id            IN VARCHAR2,
     plan_hash_value   IN NUMBER    := NULL,
     scope             IN VARCHAR2  := SCOPE_COMPREHENSIVE,
     time_limit        IN NUMBER    := TIME_LIMIT_DEFAULT,
     task_name         IN VARCHAR2  := NULL,
     description       IN VARCHAR2  := NULL )
RETURN VARCHAR2;
```

- funkce vrátí identifikátor úlohy (typicky *TASK_<číslo>*), který bude vstupem procedur pro provedení úlohy, generování zprávy a doporučeného skriptu

- **Provedení ladicí úlohy** – `EXECUTE_TUNING_TASK`

```
DBMS_SQLTUNE.EXECUTE_TUNING_TASK(
     task_name        IN VARCHAR2,
     execution_name   IN VARCHAR2 := NULL,
     execution_params IN dbms_advisor.argList := NULL,
     execution_desc   IN VARCHAR2 := NULL );
```

- procedura provede analýzu příkazu (ladicí úlohu) a umožní generování zprávy

- **Generování zprávy o ladění** – `REPORT_TUNING_TASK`

```
DBMS_SQLTUNE.REPORT_TUNING_TASK(
     task_name        IN   VARCHAR2,
     type             IN   VARCHAR2   := TYPE_TEXT,
     level            IN   VARCHAR2   := LEVEL_TYPICAL,
     section          IN   VARCHAR2   := SECTION_ALL,
     object_id        IN   NUMBER     := NULL,
     result_limit     IN   NUMBER     := NULL,
     owner_name       IN    VARCHAR2  := NULL,
     execution_name   IN  VARCHAR2    := NULL))
RETURN CLOB;
```

- tato funkce vrátí textovou zprávu v datovém typu CLOB, ve které uvádí zjištěné nedostatky při provádění příkazu a navržená řešení

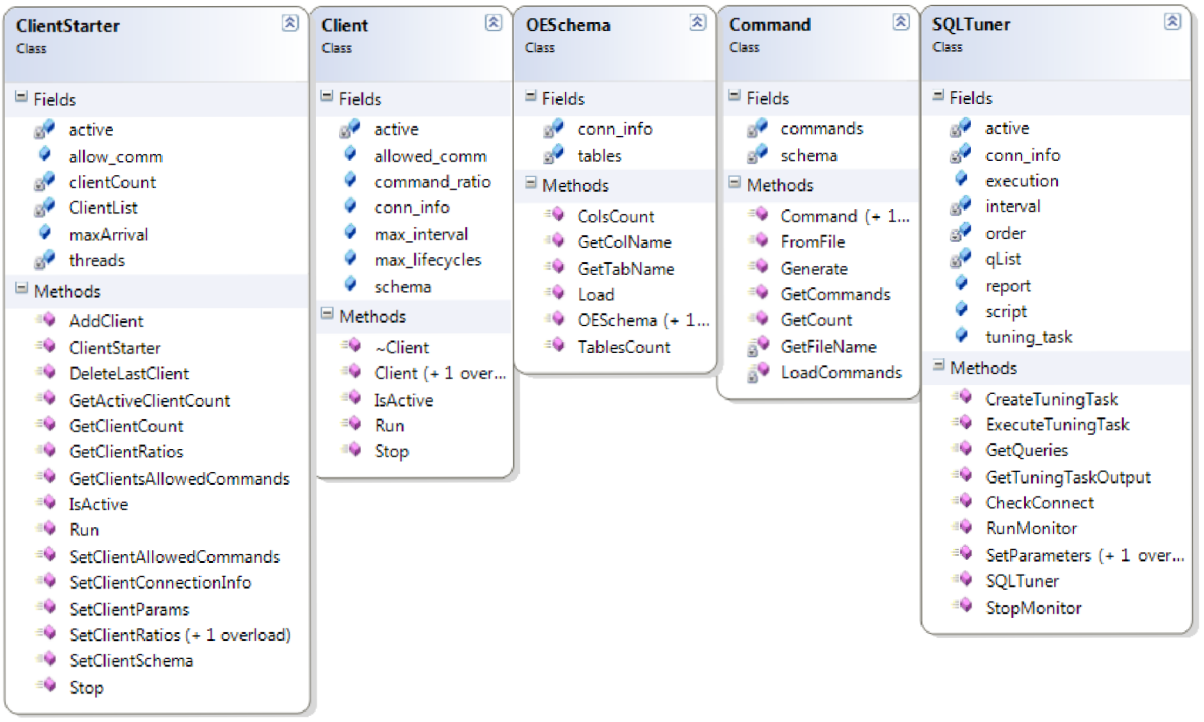
- **Generování implementačního skriptu** – `SCRIPT_TUNING_TASK`

```
DBMS_SQLTUNE.SCRIPT_TUNING_TASK(
     task_name          IN VARCHAR2,
     rec_type           IN VARCHAR2  := REC_TYPE_ALL,
     object_id          IN NUMBER    := NULL,
     result_limit       IN NUMNBER   := NULL,
     owner_name         IN VARCHAR2  := NULL,
     execution_name     IN VARCHAR2  := NULL)
RETURN CLOB;
```

- funkce vytvoří skript implementující opatření doporučená během provádění úlohy

5.5 Návrh tříd

Pro zapouzdření aplikační logiky jednotlivých rozhraní projektu jsou použity následovně definované třídy: `Client` reprezentující instanci klienta, `ClientStarter` komunikující s uživatelským rozhraním a spouštějící klienty, `OESchema` pro uchování metadat schematu databáze, `Command` pro účely generování dotazů a `SQLTuner` reprezentující instanci klienta s administrátorským oprávněním (viz. obr. 5.1).



Obrázek 5.1: Návrh tříd aplikační logiky aplikace *SQLTuningTool*

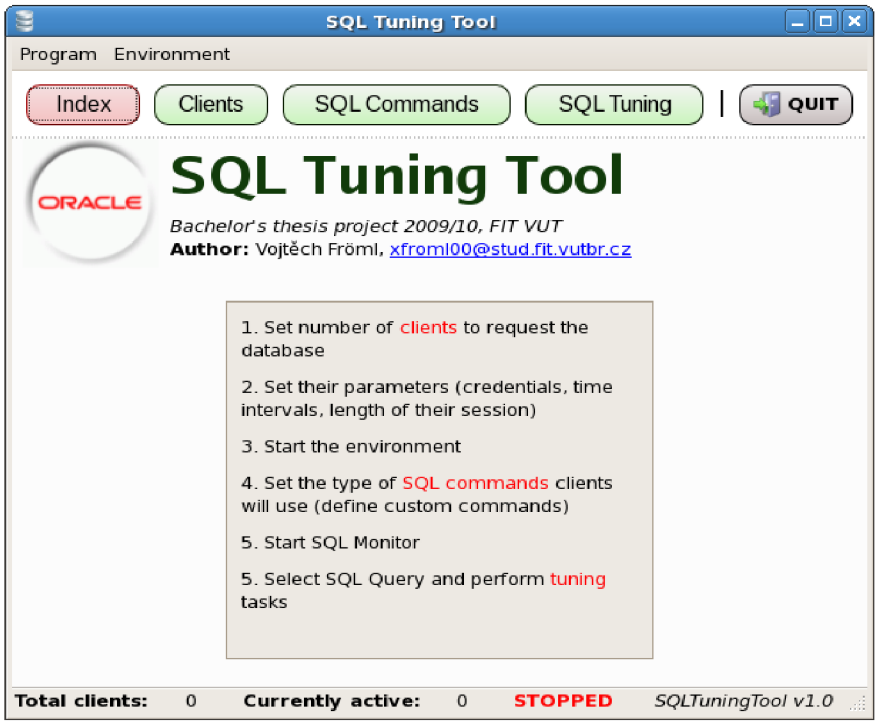
6 Implementace

6.1 Nástroje použité pro realizaci projektu

Aplikace *SQLTuningTool* byla vytvořena v programovacím jazyce C++ s využitím dynamických SQL příkazů a vestavěných bloků jazyka PL/SQL prekompilovaných nástrojem Pro*C. Grafické uživatelské rozhraní bylo votvořeno pomocí knihovny Qt 4.6.2. Vývoj aplikace a kompilace kódu proběhl v prostředí Qt Creator 1.3.1 na linuxové distribuci CentOS 5.4 32bit s nainstalovanou instancí databáze Oracle verze 11.2.0.1.0.

6.2 Uživatelské rozhraní

Návrh uživatelského rozhraní vychází z potřeby rozdělit klientské prostředí, které generuje SQL zátěž databáze a administrátorské prostředí, které ji monitoruje. Úvodní obrazovka aplikace je zobrazena na obrázku 6.1.



Obrázek 6.1: Úvodní obrazovka ukázkové aplikace

Navigace probíhá pomocí statického horního menu, které přepíná mezi obrazovkami jednotlivých prostředí, ve kterých lze nastavovat jejich lokální parametry či spouštět / zastavovat spuštěné procesy. Periodicky obnovované informace v stavovém řádku informují o aktuálním stavu klientského

prostředí. Spouštění zátěže či monitorování lze také ovládat globálně prostřednictvím programového menu.

6.3 Databázové schéma

Aplikace *SQLTuningTool* pracuje implicitně s databázovým *Sample Schema* OE. Pro její funkčnost je tedy třeba mít v databázi Oracle nainstalován balíček *Oracle Database Examples*. Aplikace spouští načítání metadat schematu do vektoru definovaných struktur tabulek `tables` třídy `OESchema` jednoduchým vestavěným blokem PL/SQL, který zároveň slouží k ověření konektivity databáze, kdykoli je to zapotřebí. Jeho část s příkazem SQL vypadá takto:

```
EXEC SQL DECLARE CURSOR cur FOR
       SELECT COLUMN_NAME FROM ALL_TAB_COLS
       WHERE TABLE_NAME = :tabName;
```

Metadata schematu slouží k automatickému generování SQL příkazů a informativně uživateli při zadávání příkazů vlastních.

6.4 Klientské prostředí

Spouštění a zastavování klientů má na starosti instance třídy `ClientStarter` běžící v samostatném vlákně po celou dobu běhu programu. Představuje mezičlánek mezi uživatelským rozhraním a instancemi třídy `Client`, reprezentujícími jednotlivé klienty. Uchovává seznam vytvořených klientů *ClientList* a pokud je je spuštěna funkcí `Run`, spouští je náhodných v intervalech od 0s do hodnoty parametru m*axArrival*. Aktualizuje jejich parametry při změně uživatelského rozhraní. To je zobrazeno na obrázku 6.2.

Pokud je instance třídy `Client` spuštěna, vygeneruje si náhodný počet životních cyklů od 0 do hodnoty *max_lifecycles*. Po ověření konektivity si inicializuje instanci třídy `Command`, předá ji svůj poměr typů SQL příkazů a ta mu vrátí pseudonáhodně generovaný či vybraný SQL příkaz.

Provedením dotazu, otevřením a uzavřením kurzoru končí jeden životní cyklus. Klient vyčká časový úsek od 0s až po hodnotu svého parametru *max_interval*. Poté se buď jeho aktivita ukončí nebo si opět vyžádá nový příkaz.

Při umělém přerušení běhu klienta nastaví instance třídy `ClientStarter` všem svým klientům seznamu hodnotu parametru *active* na *false* a vyčká na jejich ukončení aktuálně prováděného příkazu.

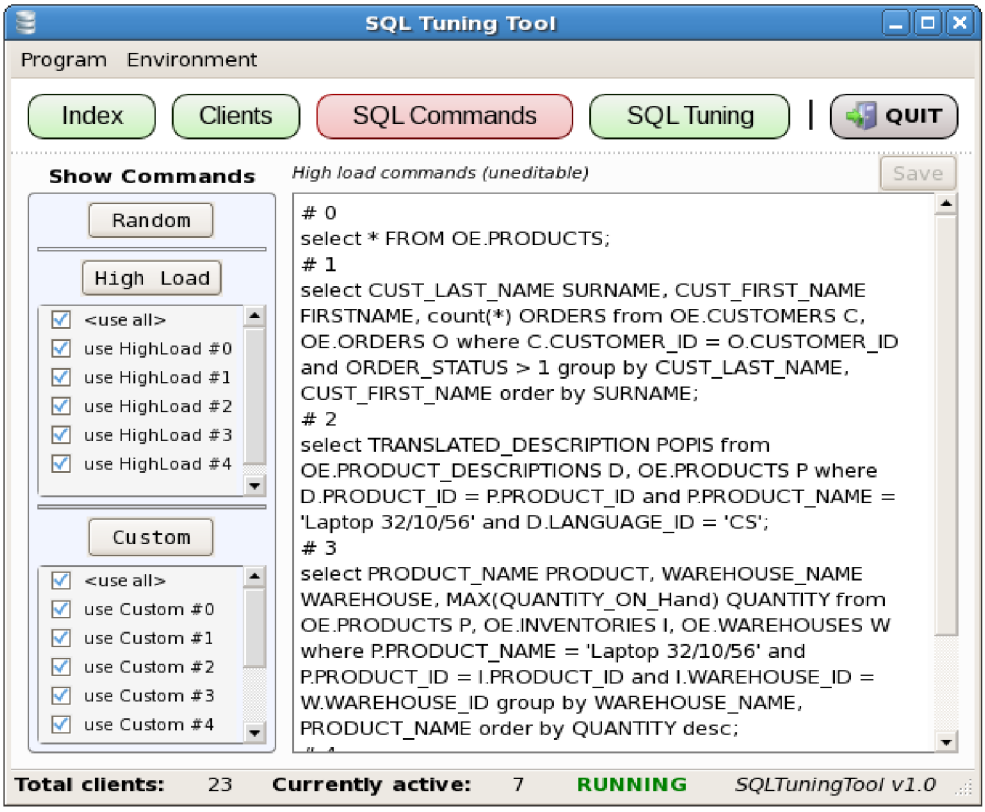
6.5 Generování SQL příkazů

Generování příkazů vyžádaných klientskými procesy zajišťuje instance třídy `Command`. Jako vstup přijímá metadata databáze ve formě instance třídy `OESchema`, strukturu *commRatio* definovanou v hlavičkovém souboru třídy `Client` určující poměr typů generovaných příkazů a názvy souborů, ze kterých načítá předdefinované příkazy.

SQL příkazy jsou rozděleny do tří kategorií:

- **Random** – náhodně generované nad schematem databáze, s proměnlivým počtem sloupců a klauzulí
- **High-Load** – předdefinované v souboru `highload.txt` uloženém v adresáři commands
- **Custom** – vytvořené uživatelem, uložené v souboru `custom.txt` tamtéž

Provádění jednotlivých předdefinovaných příkazů lze instanci třídy `Client` zakázat změněním hodnot v její struktuře *allCommands*. Editace příkazů a jejich povolování či zakazování je přístupná na obrazovce *SQL Commands* (viz. obr.6.3).



Obrázek 6.2: Editace předdefinovaných SQL příkazů

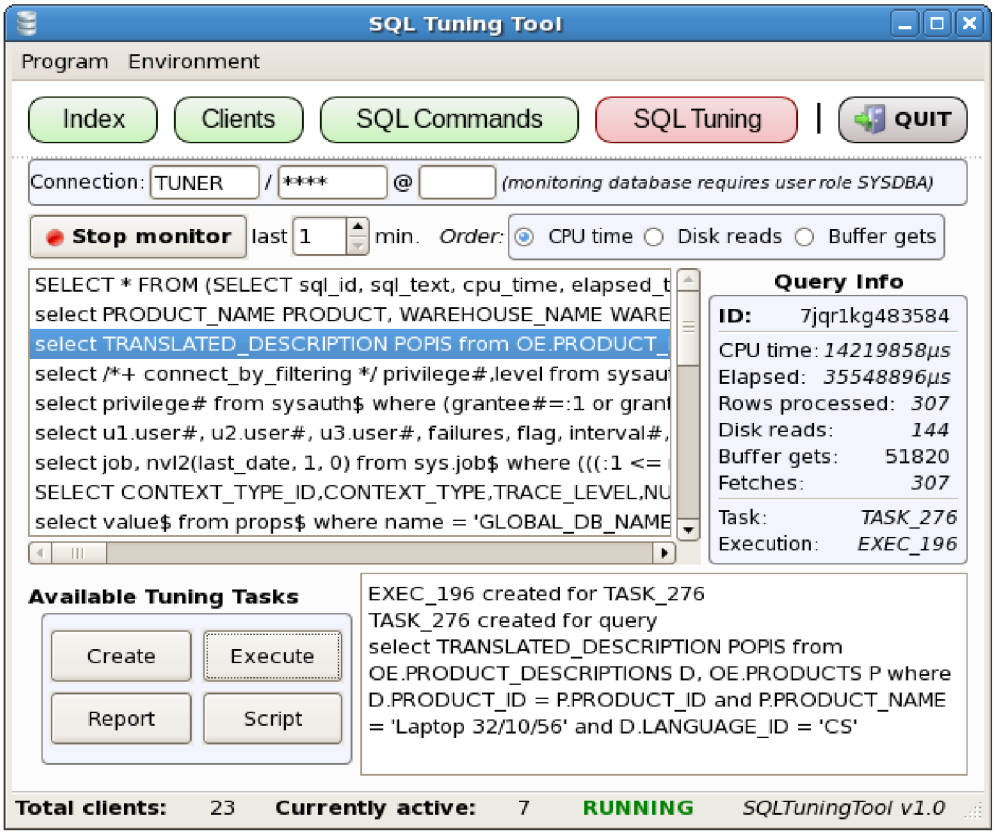
6.6 Administrátorské prostředí

Monitorování a ladění příkazů jazyka SQL je umístěno v sekci *SQL Tuning*. Po spuštění monitorování se v samostatném vlákně vytvoří instance třídy `SQLTuner`, která zajišťuje periodický sběr (každých 500ms) statistik o 25 nejnáročnějších SQL příkazech typu SELECT.

```
EXEC SQL SELECT * FROM (
     SELECT sql_id, sql_text, cpu_time, elapsed_time,
     fetches, rows_processed, disk_reads, buffer_gets
     FROM v$sql WHERE command_type = '3'
     AND last_active_time > CAST ((SELECT current time_stamp
          from dual) - interval '10' minute AS DATE)
     ORDER BY cpu_time DESC) WHERE rownum < '25';
```

Sběru statistik je možné přiřadit parametry délky monitorovaného intervalu a sloupce, podle kterého se mají data seřadit.

Po sběru první sady statistik je možné označit v seznamu položku jednoho příkazu a uložit ji tím jako aktuální hodnotu struktury *actQuery*. Uživatelské rozhraní zobrazí její parametry v infoboxu a zpřístupní tlačítka pro vytváření a provádění ladicích úkonů a následně generování zpráv a implementačních skriptů.



Obrázek 6.3: Monitorování SQL příkazů

Akce provádění automatických ladicích úloh jsou popsány v kapitole návrhu aplikace, instance třídy SQLTuner poskytuje uživatelskému rozhraní výstupy v podobě názvů úloh TASK a provádění EXECUTE (viz obr. 6. 3).

Volba *Report* nebo *Script* zobrazí výstupy procedur `REPORT_TUNING_TASK`, respektive `SCRIPT_TUNING_TASK` v textové formě. Nabídne uložení výstupu do textového souboru či skriptu `.sql` spustitelného v konzoli *SQL*Plus*.

Provedení ladicí úlohy nad samotným příkazem, kterým aplikace sbírá statistiky o SQL příkazech (viz. výše) generuje výstup *Report* s **doporučením SQL plánu** s následujícím účinkem:

```
                          Original Plan  With SQL Profile  % Improved
                          -------------  -----------------  ----------
  Completion Status:             COMPLETE           COMPLETE
  Elapsed Time(us):                26608              29255      -9.94 %
  CPU Time(us):                    19330              17197      11.03 %
  User I/O Time(us):                   0                  0
  Buffer Gets:                         0                  0
  Physical Read Requests:              0                  0
  Physical Write Requests:             0                  0
  Physical Read Bytes:                 0                  0
  Physical Write Bytes:                0                  0
  Rows Processed:                     24                 24
  Fetches:                            24                 24
  Executions:                          1                  1
```

Následuje sekce **srovnání prováděcích plánů** bez využití profilu a s profilem:

```
1- Original With Adjusted Cost
------------------------------
Plan hash value: 3011243887

-------------------------------------------------------------------------------------------
| Id  | Operation           | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                    |     1 |   597 |     3  (34)| 00:00:01 |
|*  1 |  COUNT STOPKEY      |                    |       |       |            |          |
|*  2 |   VIEW              |                    |     1 |   597 |     1 (100)| 00:00:01 |
|   3 |    SORT ORDER BY    |                    |     1 |   623 |     1 (100)| 00:00:01 |
|*  4 |     FIXED TABLE FULL| X$KGLCURSOR_CHILD  |     1 |   623 |     0   (0)| 00:00:01 |
|   5 |    FAST DUAL        |                    |     1 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(ROWNUM<25)
   2 - filter("KGLOBCLA">CAST( (SELECT /*+ PUSH_SUBQ */ CURRENT_TIMESTAMP(3)
             FROM "SYS"."DUAL" "DUAL")-INTERVAL'+00 00:01:00' DAY(2) TO SECOND(0) AS DATE))
```

```
   4 - filter("KGLOBT02"=3 AND "INST_ID"=USERENV('INSTANCE'))

2- Using SQL Profile
--------------------
Plan hash value: 1782945938

-----------------------------------------------------------------------------------------------
| Id  | Operation               | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |                   |     1 |   588 |    32   (4)| 00:00:01 |
|*  1 |  COUNT STOPKEY          |                   |       |       |            |          |
|   2 |   VIEW                  |                   |     1 |   588 |    32   (4)| 00:00:01 |
|*  3 |    SORT ORDER BY STOPKEY|                   |     1 |   623 |    32   (4)| 00:00:01 |
|*  4 |     FIXED TABLE FULL    | X$KGLCURSOR_CHILD |     1 |   623 |    29   (0)| 00:00:01 |
|   5 |      FAST DUAL          |                   |     1 |       |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(ROWNUM<25)
   3 - filter(ROWNUM<25)
   4 - filter("KGLOBT02"=3 AND "INST_ID"=USERENV('INSTANCE') AND "KGLOBCLA">CAST(
              (SELECT CURRENT_TIMESTAMP(3) FROM "SYS"."DUAL" "DUAL")-INTERVAL'+00 00:01:00'
DAY(2)
              TO SECOND(0) AS DATE))

-----------------------------------------------------------------------------
```

7 Závěr

Databázový systém Oracle 11g nabízí širokou paletu nástrojů pro správu, monitorování a ladění databáze. V této práci byly selektivně představeny některé z nich se zaměřením na monitorování výkonu databáze v reálném čase a technologie ladění příkazů jazyka SQL.

Ukázková aplikace **SQLTuningTool** umožňuje uživateli majícímu administrátorská práva k databázi Oracle nastavení, spuštění a ovládání kontrolované simulace SQL zátěže nad ukázkovými schematy. Dále poskytuje nástroj pro monitorování dopadů této zátěže na databázový výkon. Pro jednotlivé příkazy jazyka SQL pak zpřístupňuje funkce poskytované nástrojem SQL Tuning Advisor v spravovacím prostředí OEM Database Control.

Výhodou této aplikace je úplná kontrola generování SQL zátěže a z toho vyplývající možnosti testování jednotlivých příkazů jazyka SQL. Jednoduché uživatelské rozhraní v obrazovkách klienti – příkazy – ladění umožňuje rychlý přístup ke statistickým datům a zobrazení případného řešení problémů.

Další vývoj aplikace by se nepochybně ubíral směrem implementace dalších diagnostických nástrojů poskytovaných databázovým systémem Oracle 11g. Např. monitorování dlouhotrvajících příkazů jazyka SQL či paralelních příkazů v reálném čase. Dále by mohlo být zajímavé implementovat sběr statistik o aktivních relacích a jejich sledování podobným způsobem.

Testovací klientské prostředí by nepochybně šlo výrazně vylepšit přesunem nad složitější a rosáhlejší část databáze než jsou ukázková schemata. S tím souvisí i možnost pokročilejších algoritmů pro generování náročnějších náhodných příkazů.

Literatura

[1] Loney, K., Theriault M.: *Mistrovství v Oracle* Praha, Computer Press, 2002, ISBN 80-7226-635

[2] Oracle Corporation: *Oracle Database 11g: Product Editions & Features*
Dokument dostupný na URL:
<http://www.oracle.com/us/products/database/product-editions-066501.html>

[3] Hardie, W.: *Oracle Database Product Family*
Dokument dostupný na URL:
<http://www.oracle.com/technology/products/database/oracle11g/pdf/database-11g-product-family-technical-whitepaper.pdf> (říjen 2009)

[4] Bryla B., Loney K.: *Mistrovství v Oracle Database 11g* Brno, Computer Press, 2009, ISBN 978-80-251-2189-4

[5] Kolář, D.: *Pokročilé databázové systémy (prostorové databáze)*. Studijní opora, Brno, FIT VUT v Brně, 2006

[6] Oracle Corporation: *Oracle Enterprise Manager*
Dokument dostupný na URL:
<http://www.oracle.com/us/products/database/database-manageability-066512.html>

[7] Wu C.: *Oracle Enterprise Manager Blog*
Dokument dostupný na URL:
<http://blogs.oracle.com/oem/2010/04/oracle_enterprise_manager_11g_2.html> (duben 2010)

[8] Oracle Corporation: *Introducing Oracle Enterprise Manager 11g*
Dokument dostupný na URL:
<http://www.oracle.com/us/products/enterprise-manager/oem11g-integrated-it-mgmt-strategy-069138.pdf>

[9] Oracle Corporation: *Oracle Enterprise Manager Grid Control Extensions Exchange*
Dokument dostupný na URL:
<http://www.oracle.com/technology/products/oem/extensions/index.html>

[10] Wikipedia.org: *Oracle Enterprise Manager*
Dokument dostupný na URL:
<http://en.wikipedia.org/wiki/Oracle_Enterprise_Manager> (duben 2010)

[11] Oracle Corporation: *Oracle Database Documentation Library 11g Release 2*
Dokument dostupný na URL:
<http://www.oracle.com/pls/db112/homepage>

[12] Oracle Corporation: *Oracle Diagnostic Pack*
Dokument dostupný na URL:
<http://www.oracle.com/us/products/enterprise-manager/diagnostic-pack-11g-ds-068465.pdf>

[13] Hall, T.: *Automatic Workload Repository in Oracle Database 11g*
Dokument dostupný na URL:
<http://www.oracle-base.com/articles/10g/AutomaticWorkloadRepository10g.php>

[14] Belknap P., Koltakov S., Raitto S.: *Advanced SQL Tuning Features of Oracle Database 11g*
Dokument dostupný na URL:
<http://www.oracle.com/technology/products/manageability/database/pdf/ow07/sqltune_presentation_ow07.pdf> (listopad 2007)

[15] Mintz S., Norris C., Cohen B.: *Comparative Management Costs Study - Oracle Database 11g vs. Microsoft SQL Server 2008*
Dokument dostupný na URL:
<http://www.oracle.com/us/products/database/039433.pdf> (červenec 2009)

[16] Rittman, M.: *Enterprise Manager Grid Control For Windows Now Available*
Dokument dostupný na URL:
<http://www.rittmanmead.com/2004/08/23/enterprise-manager-grid-control-for-windows-now-available/> (duben 2004)

Seznam příloh

Příloha A. Uživatelská příručka

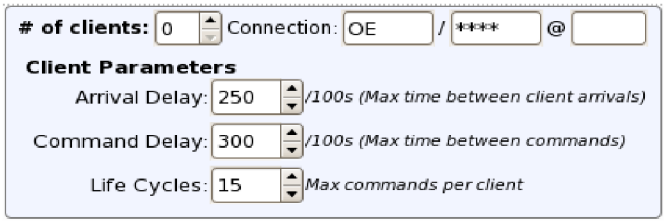
Příloha B. Obsah CD

Příloha A

Uživatelská příručka

Aplikaci SQLTuning Tool můžete spustit na unixových platformách 32bit binárním souborem z adresáře `build` na přiloženém CD. K běhu je nutné zachovat adresářovou strukturu nadřazené složky, zejména složky knihoven `lib` a složky příkazů `commands`.

I. Pro spuštění klientského prostředí musíme mít vytvořený účet na existující instatnci databázi Oracle. Prvním krokem po zobrazení obrazovky **CLIENTS** je zadání správných přihlašovacích údajů do polí *connection* a nastavení globálních parametrů klientského prostředí **Client parameters**:

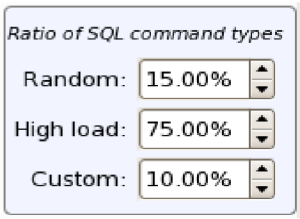


Obrázek A.1: Nastavení globálních parametrů klientského prostředí

- *Arrival Delay* – maximální interval mezi připojováním jednotlivých klientů
- *Command Delay* – maximální prodleva mezi příkazy jednoho klienta
- *Life Cycles* – maximální počet příkazů jednoho klienta

Tyto parametry lze měnit i za spuštěného prostředí bez potřeby jej zastavovat.

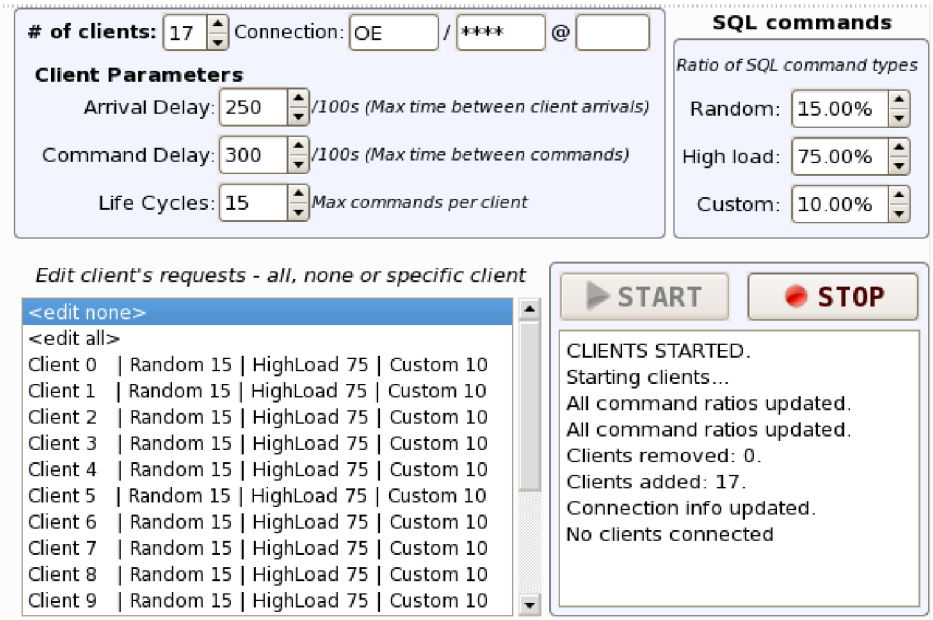
Ještě před zadáním celkového počtu klientů do pole **# of clients** nastavíme v boxu **SQL commands** poměr mezi typy příkazů, které bude klient posílat databázi. Detailní nastavení pak poskytuje 3. obrazovka *SQL commands* (viz. dále). Takto nastavený poměr bude aplikován na všechny nově vytvořené klienty.



Obrázek A.3: Nastavení poměru typů SQL příkazů

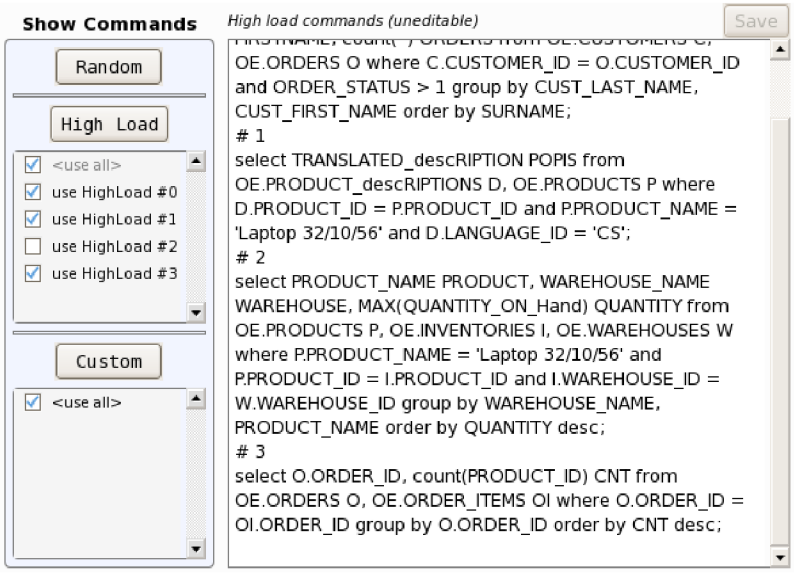
Můžeme pomocí pole **# of clients** přidat klienty, jejich přehled vidíme v seznamu klientů. Můžeme spustit prostředí tlačítkem **START.**

Jednotlivé globální i klientské parametry lze měnit i za běhu prostředí, veškeré údaje stačí potvrzovat stisknutím tlačítka Enter. V pravém dolním infoboxu by se následně měla objevit informace o změně.



Obrázek A.4: Spuštění klientského prostředí

II. Chceme-li se podrobněji zabývat nastavením příkazů jazyka SQL, postoupíme na obrazovku **SQL COMMANDS**. Ovládacím menu v levém sloupci si můžeme zobrazit jednotlivé příkazy v hlavním textovém poli. Odškrtnutím checkboxu vyřadíme příkaz s odpovídajícím číslem z provádění. Příkazy typu Custom lze navíc editovat, takto provedené změny je třeba uložit tlačítkem *Save*.



Obrázek A.5: Editace příkazů

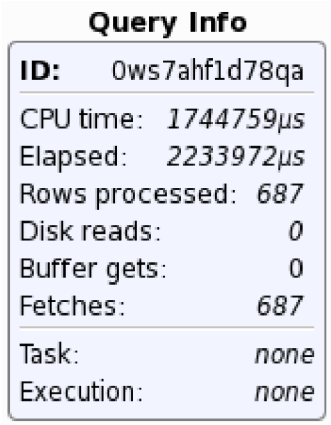
III. Ke sledování výkonu SQL příkazů na stránce **SQL Tuning** musíme zadat připojení k databázi s oprávněním DBA. Míra oprávnění je automaticky kontrolována při spuštění monitorování. Po

úspěšném připojení se každých 250ms obnovuje seznam příkazů, který můžeme seřadit podle třech různých kritérií:

- **Cpu reads** – procesorový čas strávený prováděním příkazového plánu
- **Disk reads** – počet fyzických přístupů k disku
- **Buffer gets** – počet nálezů ve vyrovnávací paměti

Dalšími parametry dotazu jsou:

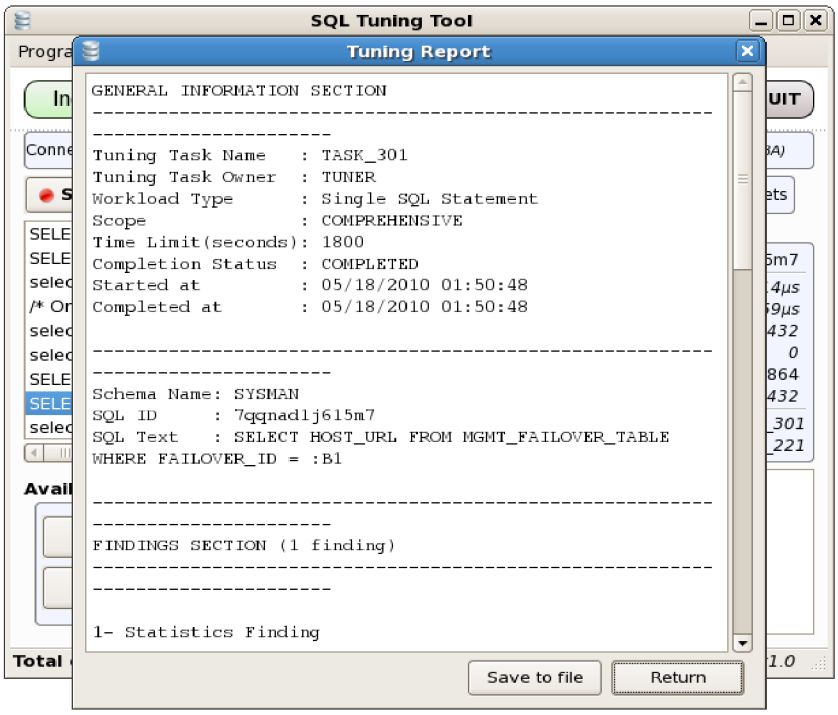
- **ID** – identifikátor příkazu
- **Elapsed** – celkový čas strávený vyřizováním příkazu
- **Rows processed** – počet zpracovaných řádků
- **Fetches** – počet operací získání dat



Obrázek A.6: Detail SQL příkazu

Ladění dotazů probíhá pomocí automatických nástrojů, je možné jej aktivováním postupně 4 fází automatického ladění:

- **Vytvoření** a **provedení ladicí úlohy**
- **Generování zprávy** a její případný export do souboru *Save to file*
- **Vytvoření** a následná manuální implementace opravného **skriptu**



Obrázek A.7: Výstup zprávy Report

Příloha B

Obsah CD

`/build`

spustitelný binární soubor `SQLTuningTool`

`/commands`

soubory `highload.txt` a `custom.txt` s předdefinovanými příkazy

`/lib`

knihovny potřebné pro spuštění, případně překlad zdrojových kódů

`/src`

zdrojové kódy, soubor Makefile

`/src/env`

zdrojové kódy aplikační logiky

`/src/gui`

zdrojové kódy tříd uživatelského rozhraní

`Nastroje_pro_spravu_a_ladeni_Oracle_11g.pdf`

technická zpráva ve formátu pdf

`Nastroje_pro_spravu_a_ladeni_Oracle_11g.odt`

technická zpráva ve formátu OpenDocument