UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Systém pro sledování činnosti uživatelů databázového serveru

Helena Zechmeisterová

Bakalářská práce
2014

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Helena Zechmeisterová**
Osobní číslo:	**I11182**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Systém pro sledování činnosti uživatelů databázového serveru**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je navrhnout řešení, pomocí kterého by bylo možné sledovat činnosti uživatelů na databázovém serveru Oracle. Systém bude umožňovat ihned po přihlášení uživatele sledovat všechny jeho činnosti na databázovém serveru. Všechny informace se budou uchovávat na tomtéž serveru. Výsledná data budou k dispozici v prostředí webové aplikace. Použité technologie, HMTL5, PHP a databáze Oracle.
V teoretické části bude provedena rešerše v oblasti zmapování novinek databázového serveru Oracle 12c.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. Lacko, L.: Oracle - Správa, programování a použití databázového systému. Computer Press, 2007.ISBN 978-80-251-1490-2.
2. Kyte, T.: Oracle - Návrh a tvorba aplikací. Computer Press, 2006. ISBN 80-251-0569-5.
3. LUBBERS, Peter, Brian ALBERS a Frank SALIM. HTML5: programujeme moderní webové aplikace. Vyd. 1. Brno: Computer Press, 2011, 304 s. ISBN 978-80-251-3539-6.
4. Oracle Database 11g release 2. Oracle database Documentation library [online]. Oracle America, Inc., 2013 [cit. 2013-10-31]. Dostupné z: http://www.oracle.com/pls/db112/homepage

Vedoucí bakalářské práce: **Ing. Tomáš Váňa**
Katedra informačních technologií

Datum zadání bakalářské práce: **20. prosince 2013**
Termín odevzdání bakalářské práce: **9. května 2014**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2014

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byla jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 6. 5. 2014 Helena Zechmeisterová

Poděkování

Ráda bych touto cestou poděkovala panu Ing. Tomášovi Váňovi za výbornou spolupráci při tvorbě této práce. V neposlední řadě rodině, přátelům a hlavně manželovi za podporu a pomoc po celou dobu studia.

Anotace

Práce se zaměřuje na možnosti získávání informací o činnosti uživatelů na databázovém serveru Oracle. Konkrétně popisuje principy (AWR, FGA, Library cache), které následně využívá aplikace, která vznikla jako součást této práce. Teoretická část je zaměřena na zmapování novinek, které jsou k dispozici v nové verzi databázového systému Oracle 12c.

Klíčová slova

oracle, oracle 12c, novinky, AWR, FGA, library cache

Title

The system for monitoring of database user's behavior.

Annotation

This bachelor thesis deals with gathering of the information about database Oracle user’s behavior. Talks about principles which are used in application (AWR, FGA, Library cache). The theory part is aimed to map news in database system Oracle 12c.

Keywords

oracle, oracle 12c, news, AWR, FGA, library cache

Obsah

Seznam zkratek

LC	Library Cache
SGA	Systém Global Area
AWR	Automatic Workload Repository
FGA	Fine Grained Auditing
LRU	Last Recently Used
DDL	Data Definition Language
ACID	Atomicity, Consistenci, Isolation, Durability
LRU	Least Recently Used
ASH	Active Session History
OLE	Object Linking and Embedding
ADO	ActiveX Data Objects

Seznam obrázků

Seznam tabulek

Úvod

Databázový systém Oracle poskytuje velké množství informací o činnosti uživatelů. Informace, které jsou k dispozici, obsahují mimo jiné datum a čas navázání spojení, ukončení spojení. Dále také informace o provedených dotazech a o vytvořených databázových objektech.

Cílem této bakalářské práce je vytvořit webovou aplikaci, která získává a zobrazuje informace o činnosti uživatelů v prostředí databázového serveru Oracle 11g2. Tato aplikace vznikla primárně pro výukové účely databázových systémů. Aplikace umožňuje zobrazení základních informací o uživateli, jeho napojení na databázové předměty a přehled dotazů, které uživatel spouštěl. Dotazy jsou k dispozici jednak z aktuální cache paměti, z auditu FGA, ale také z AWR reportů.

Teoretická část práce obsahuje přehled novinek, které obsahuje nově vydaná verze databázového systému Oracle 12c. Do práce bylo vybráno pouze několik nejzajímavějších změn oproti minulé verzi. V následující části jsou popsané principy, kterými je možné sledovat činnosti uživatelů databázového serveru. Zejména jsou zde uvedené ty způsoby, které byly použity v praktické části práce. Závěrečná část práce se zabývá samotnou aplikací, který byla vytvořena.

Uživatelé mohou přistupovat k této aplikaci ve 4 režimech. První režim je nepřihlášený uživatel, který nemá téměř žádné pravomoci z důvodu důvěrnosti dat. Druhý režim je přihlášení běžného uživatele, který může vidět pouze svá data, pokud má účet spojený s účtem v databázi. Další stupeň zabezpečení představuje přihlášení správce skupiny, který může sledovat uživatele v rámci připojení k jeho skupině. Posledním 4. režimem je administrátor, kterému náleží všechna práva. Přihlášení do aplikace je odstíněno od přihlášení do databáze. Uživatel, který získá přístup do databáze, nemusí nutně získat přístup i do aplikace.

1 Oracle Database 12c

Nová verze databázového systému Oracle Database 12c (c = Cluster) představuje velký skok kupředu v oblasti databázových systémů. Nová verze s sebou přináší spoustu novinek, některé vybrané budou postupně v této kapitole představené, a srovnané se starší verzí.

1.1 Novinky ve verzi 12c

Následující kapitola bude rozebírat novinky a vylepšení, které vyšly v nové verzi databázového systému Oracle 12c. Počet těchto změn se počítá na stovky a jejich popis by vydal na samostatnou knihu. Cílem této kapitoly je vybrat některé zajímavé novinky, které dle mého názoru, patří k těm nejzajímavějším a týkají se právě této práce. Pro kompletní seznam všech novinek doporučuji prostudovat dokumentaci, ze které jsem čerpala v první řadě. Dokumentace je k dispozici na adrese: http://docs.oracle.com/cd/E16655_01/server.121/e17209/release_changes.htm#SQLRF56314. Nyní se zaměříme na konkrétní změny, které jsem vybrala jako nejzajímavější.

1.1.1 Rozšíření funkce sys_context

Funkce sys_context vrací informace z kontextu aktuální session. Informace jsou uloženy ve jmenných prostorech, které jsou dva. Prvním je USERENV a druhý SYS_SESSION_ROLES. První jmenovaný obsahuje informace o aktuálním sezení jako je uživatelské jméno, stanice, uživatelské jméno operačního systému atp. Druhý jmenný prostor obsahuje informace o rolích, které jsou v rámci session k dispozici.

Ke změně oproti minulé verzi došlo v rozšíření parametrů, které jmenný prostor USERENV poskytuje. V následující tabulce je k dispozici přehled nových parametrů (1).

- **CDB_NAME** – jméno CDB[1], pokud to není multitenant container database[2] tak NULL.
- **CLIENT_PROGRAM_NAME** - jméno klientského programu.
- **CON_NAME** - jméno containeru, pokud to není multitenant databáze tak DB_NAME.
- **DB_SUPPLEMENTAL_LOG_LEVEL** – vrací level v případě supplemental loggingu, jinak null. Možné hodnoty jsou stejné jako supplemental loggingu[3], tedy ALL_COLUMN, FOREIGN_KEY, MINIMAL, PRIMARY_KEY, PROCEDURAL a UNIQUE_INDEX.
- **IS_APPLY_SERVER** - pro Oracle Data Guard – true na apply server, jinak false.

[1] Current Database – aktuální databáze
[2] http://docs.oracle.com/cd/E16655_01/server.121/e17633/cdbovrvw.htm#CNCPT89234
[3] http://docs.oracle.com/cd/E11882_01/server.112/e22490/logminer.htm#SUTIL1582

- **ORACLE_HOME** – vrací ORACLE_HOME.
- **PLATFORM_SLASH** - oddělovač adresářové cesty na dané platformě.
- **SHEDULER_JOB** - Y = je to spuštěno dbms_scheduler, N – není to spuštěno v dms_scheduler. (2).

1.1.2 Definování PL/SQL funkcí přímo v SQL

Databázový systém Oracle podporuje vytváření uživatelských funkcí a procedur. Tyto fragmenty kódu se vytváří v procedurálním jazyce PL/SQL. Vytvořené funkce je možné použít přímo v SQL dotazu. S novou verzí 12c je však možné tyto funkce definovat přímo v SQL a to za klauzulí WITH.

Standardní klauzule WITH umožňuje vytvoření pojmenované části dotazu (inline pohled), který se dá potom použít v další části dotazu, viz ukázka (2).

```
with pojmenovano as
    (select dummy as my_dummy from dual)
select
    my_dummy
from pojmenovano;
```

Nyní je možné tuto klauzuli využít i pro definici části PL/SQL kódu.

```
with function with_result(x varchar2) return varchar2 is
begin
    return x|| ' - x';
end;

select
    with_result(dummy) as with_result_on_dummy
from dual;
```

Pokud chceme tuto možnost využít i v DML operacích (INSERT, UPDATE, DELETE nebo MERGER), musíme použít nový hint[4] WITH_PLSQL. Použití tohoto hintu je potom následující (3):

```
update /*+ with_plsql */ tsttbl x
set col2 = (with function fn_useless_wait(p_seed in number
default 5) return varchar2 is
  begin
     dbms_lock.sleep(p_seed);
     dbms_output.put_line(rpad('z', p_seed, 'z'));
     return dbms_random.string('a', p_seed);
  end fn_useless_wait;
select fn_useless_wait(x.col1) from dual);
```

[4] Speciální komentář, na základě kterého je vývojář schopen ovlivnit zpracování SQL

1.1.3 Adaptivní exekuční plán

Do verze 12c fungoval optimalizátor (CBO) tak, že sestavil exekuční plán na základě předpokladů (statistik) a tímto plánem se musel držet. Pokud například dotaz prováděl spojení tří tabulek, kde z první tabulky šly 3 řádky, z druhé tabulky 5 a ze třetí 10 řádků, optimalizátor pravděpodobně zvolí jako metodu spojení nested loop. Pokud by však řádků bylo řádově 1000 krát více, jako lepší metodu by zvolil hash join. Pokud k takovému rozporu oproti odhadu došlo dříve, optimalizátor nemohl na nastalou situaci nijak reagovat a musel se řídit stávajícím exekučním plánem.

Při použití adaptivního exekučního plánu jsou v okamžiku sestavování exekučního plánu tyto plány vytvořeny rovnou dva. V okamžiku, kdy dojde k zásadnímu rozporu mezi odhadem a realitou (zjištěno pomocí on-line sběru statistik), může se dotaz dále provádět dle alternativního exekučního plánu (optimalizátor se adaptuje na novou situaci).

Nyní si ukážeme použití adaptivního exekučního plánu na příkladu. Nejdříve se ujistíme, že máme nastavené potřebné parametry:

```
select name, value from v$parameter where name like
'optimizer_f%' or name like '%adaptive_r%';
```

NAME	VALUE
optimizer_features_enable	12.1.0.1
optimizer_adaptive_reporting_only	FALSE

Využití adaptivních plánů budeme demonstrovat na následujícím jednoduchém dotazu:

```
select product_name
  from oe.order_items o, oe.product_information p
  where o.unit_price = 15
  and o.quantity > 1
  and p.product_id = o.product_id;
```

Pokud si zobrazíme exekuční plán toho příkazu bez jeho spuštění (příkaz EXPLAIN PLAN FOR), dostaneme plán, který dle CBO bude optimální pro spuštění.

```
-----------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                      |     4 |   128 |     7   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                |                      |       |       |            |          |
|   2 |   NESTED LOOPS               |                      |     4 |   128 |     7   (0)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL         | ORDER_ITEMS          |     4 |    48 |     3   (0)| 00:00:01 |
|*  4 |    INDEX UNIQUE SCAN         | PRODUCT_INFORMATION_PK |     1 |       |     0   (0)| 00:00:01 |
|   5 |   TABLE ACCESS BY INDEX ROWID| PRODUCT_INFORMATION  |     1 |    20 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------
```

Z plánu je patrné, že jako optimální byla zvolena metoda spojení nested loops, která je

optimální pro menší množiny dat a s výhodou se využívá v kombinaci s indexem. Nyní se podíváme, jaký exekuční plán se ve skutečnosti provede, pokud zobrazený dotaz spustíme.

Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem
0	SELECT STATEMENT		1		13	00:00:00.01	23			
* 1	HASH JOIN		1	4	13	00:00:00.01	23	2061K	2061K	411K (0)
- 2	NESTED LOOPS		1		13	00:00:00.01	7			
- 3	NESTED LOOPS		1	4	13	00:00:00.01	7			
- 4	STATISTICS COLLECTOR		1		13	00:00:00.01	7			
* 5	TABLE ACCESS FULL	ORDER_ITEMS	1	4	13	00:00:00.01	7			
- * 6	INDEX UNIQUE SCAN	PRODUCT_INFORMATION_PK	0	1	0	00:00:00.01	0			
- 7	TABLE ACCESS BY INDEX ROWID	PRODUCT_INFORMATION	0	1	0	00:00:00.01	0			
8	TABLE ACCESS FULL	PRODUCT_INFORMATION	1	1	288	00:00:00.01	16			

Exekuční plán, který byl skutečně proveden, se liší od předpokládaného. Tudíž byl zvolen alternativní plán. Důvod pro jeho zvolení je vidět na řádku 8, kde optimalizátor předpokládal, že bude pracovat s jednořádkovou množinou (E-Rows). Namísto toho byla množina o četnosti 288 řádků (A-Rows). Pro tento počet řádků již použití nested-loops není výhodné a je výhodnější využití hash join. Exekuční plán byl proto upraven a použit byl alternativní exekuční plán. Operace, které byly uvedeny v původním exekučním plánu a v alternativním ne, jsou označeny znaménkem -.

1.1.4 White list

Pod označením White list je skryta klauzule ACCESIBLE BY, kterou je nyní možné použít pro PL/SQL procedury a funkce. Tato klauzule nám specifikuje, odkud je možné danou proceduru/funkci spustit.

Tato klauzule je vhodná zejména pro testování a vývoj samotný, protože odpadá problém, kdy dojde ke spuštění nějaké procedury bez potřebných návazností (například bez zápisu do logu) (2).

Použití této klauzule je velice jednoduché, viz příklad:

```
create or replace procedure prc_print_ha accessible by
(procedure prc_call_with_log) as
begin
    dbms_output.put_line('Zde by se upravoval zákazník');
end;
```

Příklad představuje definici procedury, kterou je možno spustit pouze z procedury prc_call_with_log (2).

1.1.5 Automatické generování UID

Předchozí verze databázového systému nepodporovala automatické generování hodnot, například pro primární klíče. Tuto funkci zastávaly sekvence. Do nové verze 12c byl přidán sloupec typu IDENTITY, který tuto možnost poskytuje (4).

Na pozadí této funkcionality se ale stále nachází sekvence, jenom je její použití pro uživatele transparentní. Příklad vytvoření sloupce typu identity může vypadat následovně (5):

```
CREATE TABLE t1 (id NUMBER GENERATED AS IDENTITY);
```

Jelikož je na pozadí sekvence, tak i při definování sloupce je možné definovat rozsah dané sekvence (5):

```
CREATE TABLE t2 (id NUMBER GENERATED BY DEFAULT AS IDENTITY
(START WITH 100 INCREMENT BY 10));
```

Princip fungování je patrný i z exekučního plánu DML dotazu. V exekučním plánu (Obrázek 1) je uvedený záznam o načtení nové hodnoty ze sekvence.

```
INSERT INTO identity_test_tab (description) VALUES ('Just DESCRIPTION');

Execution Plan
----------------------------------------------------------
Plan hash value: 993166116

-------------------------------------------------------------------------------------------------
| Id  | Operation                 | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT          |                   |     1 |   100 |     1   (0)| 00:00:01 |
|   1 |  LOAD TABLE CONVENTIONAL  | IDENTITY_TEST_TAB |       |       |            |          |
|   2 |   SEQUENCE                | ISEQ$$_92117      |       |       |            |          |
-------------------------------------------------------------------------------------------------
```

Obrázek 1 - Exekuční plán při vkládání do identity sloupce

1.1.6 Top-N query

Top-N query je označení pro dotaz, ve kterém je limitovaný výstup na prvních n záznamů. Podobnou limitaci zná spousta uživatelů z prostředí databázového systému MySQL, kde existuje klauzule LIMIT. V Oraclu byla tato funkcionalita realizována pomocí pseudosloupce ROWNUM, který vrací čísla řádků výsledku.

Nová verze databázového serveru Oracle přinesla zjednodušení. K dispozici jsou nové dvě klauzule a to OFFSET a FETCH. Syntaxe obou klauzulí je uvedena níže.

```
[ OFFSET offset { ROW | ROWS } ]
[ FETCH { FIRST | NEXT } [ { rowcount | percent PERCENT } ]
    { ROW | ROWS } { ONLY | WITH TIES } ]
```

```
select first_name from hr.employees order by first_name fetch
first 4 rows with ties;
```

Prvním příkladem z tabulky employees chceme první 4 křestní jména po abecedním seřazením. Doplnění klauzule with ties znamená, že ve výsledku se může objevit i více řádků, ale pouze 4 unikátní hodnoty. Pro lepší představu je zobrazený výsledek dotazu, ze kterého je patrné, že ačkoliv očekáváme pouze 4 řádky, výstupem je řádků 5, protože jméno Alexander se vyskytuje v tabulce dvakrát.

	FIRST_NAME
1	Adam
2	Alana
3	Alberto
4	Alexander
5	Alexander

```
select first_name from hr.employees order by first_name fetch
first 25 percent rows only;
```

Druhý příklad využívá procentuální vyjádření. Z tabulky tedy chceme prvních 25 % záznamů, po abecedním seřazení. Výsledkem tedy bude celkem 27 z celkových 107 možných řádků.

1.1.7 Optimalizace práce s úložištěm

Verze 12c poskytuje vylepšené řešení pro správu úložiště. Pro lepší představu o využívání datových bloků si pro každý segment systém vytváří heat mapu. Bloky, které jsou využívané často, jsou označené jako teplejší a bloky, se které se téměř nenačítají, jsou označené jako chladnější. Na základě heat mapy je následně rozhodnuto, které bloky se budou komprimovat s větším kompresním poměrem (6).

1.1.8 Vylepšené statistiky

Statistky jsou v nové verzi vylepšené výrazným způsobem. Za zmínku stojí hlavně fakt, že frekvenční histogramy již nejsou limitované počtem 256 hodnot. Na starších verzích nebylo možné využít frekvenční histogram na sloupec, kde počet unikátních hodnot překročil 256. Pro tyto sloupce bylo možné použít pouze high-balanced histogramy.

Další podstatné vylepšení ve statistikách spočívá v tom, že statistiky jsou sbírané on-line přímo při nahrávání dat do databáze. Dosud při nahrávání většího množství dat bylo nutné ručně spustit sběr statistik, nebo se muselo čekat, většinou do večera, kdy standardně probíhá sběr statistik.

1.1.9 Application continuity

Application continuity (AC) je funkcionalita, která je pro real application cluster. Jeho fungování spočívá v tom, že v okamžiku kdy dojde k přerušení spojení mezi aplikací a jedním uzlem, je celá transakce přenesena on-line na jiný uzel bez vědomí uživatele. Stejná funkce existuje i ve straších verzích, ale bezchybný přenos byl možný pouze pro transakce, které byly pro čtení. Transakce, které byly pro čtení a zápis, nebylo možné obnovit na novém uzlu. Nová verze AC je již schopná přenášet i transakce v režimu čtení/zápis (7).

Další funkcí, která zlepšuje komfort uživatelských aplikací je transaction guard. Tato funkce nepoví aplikaci, aby provedla například dvakrát odeslání objednávky atp.

1.1.10 Ostatní novinky

Mezi ostatní novinky, které stojí za zmínku, ale nejsou tak výrazné, aby vydaly na vlastní odstavec.

- Kapacita datových typů VARCHAR2, CHAR a RAW byla rozšířená na kapacitu 32k ze stávajících 4k.
- Pro dočasné tabulky je undo generováno do dočasného tabulkového prostoru. Dočasná data se tedy nemíchají k ostatním stálým datům.

- K velkému pokroku došlo v oblasti partitionigu, kde je nyní možné provádět DDL operace nad více partition současně. Dále je také umožněno připojovat a odpojovat partition on-line za běhu.
- Globální indexy nemusí být přebudovány v okamžiku změny partiton. To znamená, že propojení přes tato data může být stále k dispozici, i když jsou data již smazaná.
- Příkaz TRUNCATE má nyní nově klauzuli CASCADE, která umožní kaskádní smazání závislých záznamů.
- Datové soubory je nyní možné přesouvat a přejmenovávat on-line, tedy za běhu databáze.
- Defaultní hodnota sloupce nyní může být i ze sekvence. Dosud tato varianta nebyla možná, i když se nabízela jako logická.

1.2 Odstraněné funkce

Nová verze databázového systému Oracle 12c přinesla nejen spoustu nových funkcí, ale také spousta funkcí přestala být podporovaná v této verzi. Tato kapitola tvoří jenom přehled některých funkcí, které již dále nebudou v Oraclu podporované, nebo jsou označené jako zastaralé a jejich podpora skončí v následující verzi. Kompletní seznam je možné nalézt v dokumentaci (viz zdroj (8)).

- Oracle Enterpise Manager Database Control je nahrazený nástrojem Oracle Enterprise Manager Express.
- OLE Objekty již nadále nejsou podporované a v aplikacích je třeba přejít na ADO, případně jiné technologie.
- Ve verzi 12c jsou jako zastaralé označeny některé parametry. Celkový výčet zastaralých (deprecated) paramterů je možný po spuštění příkazu:

```
SELECT name from v$parameter
          WHERE isdeprecated = 'TRUE' ORDER BY name;
```

Odstraněných nebo zastaralých funkcí a parametrů je obrovské množství. Každému administrátorovi databázového serveru doporučuji, aby si před přechodem na novou verzi prostudoval dokumentaci (8).

Samotné nasazení nové verze do ostrého produkčního provozu bude ještě nějakou dobu trvat, protože některé změny jsou tak zásadní, že standardní přechod bez zásahu do aplikací není možná.

2 Získávání informací z databázového serveru

Databázový server Oracle nám poskytuje spoustu možností, jak sledovat činnosti uživatelů. Tyto možnosti jsou dané tím, že sám server je do velké míry samodokumentující. To znamená, že většina činností, které server provádí sám, nebo které na něm provádí uživatelé, je logováno. Některé činnosti jsou logovány z hlediska bezpečnosti nebo konzistence dat (dodržení ACID), a některé jsou k dispozici pro případ, že by jich bylo třeba. Některé z těchto činností server dokumentuje automaticky, některé se musí explicitně zapnout. Problém při auditu spočívá v tom, že může docházet z extrémnímu nárůstu dat a tedy tabulkových prostorů. Tento problém je třeba řešit a jeho řešením se budeme zabývat dále v této kapitole.

Pro získávání informací je nutné pracovat pod vysokým stupněm oprávnění (DBA, SYSDBA). Běžný uživatel se k těmto datům nesmí dostat.

2.1 Sledování přihlašování uživatelů

Sledování uživatelů je možné jednak implicitně pomocí auditu, nebo vlastní cestou pomocí triggeru. V této kapitole si popíšeme obě zmíněné možnosti.

2.1.1 DBA_AUDIT_TRAIL

Jednoduché a pro databázový server nativní. Netřeba explicitně spouštět. Je třeba hlídat jen velikost tabulkových prostorů. Pro zobrazení těchto informací musí mít uživatel práva na úrovni DBA (9).

Dotazem do tohoto pohledu můžeme mimo jiné zjistit následující informace (9):

OS_USERNAME – uživatelské jméno operačního systému, ze kterého se k databázi přistupuje.
USERNAME – uživatelské jméno do databáze.
USERHOST – název uživatelského počítače.
ACTION_NAME – název akce, kterou záznam představuje (přihlášení, odhlášení atp.).
TIMESTAMP – časové razítko výskytu události.
OS_PROCCESS – název procesu operačního systému, který k databázi přistupuje.

Kompletní seznam je možné nalézt v dokumentaci Oraclu (viz zdroj (9)).

2.1.2 Použití triggeru

Pokud nám stačí pouze základní informace, můžeme si připravit trigger, který se bude spouštět po přihlášení uživatele k databázovému serveru a zapíše požadované informace do tabulky. Pro zjištění základních informací můžeme s výhodou využít funkci sys_context, která nám může vrátit informace ze jmenného prostoru USERENV. Základní struktura triggeru může vypadat následovně:

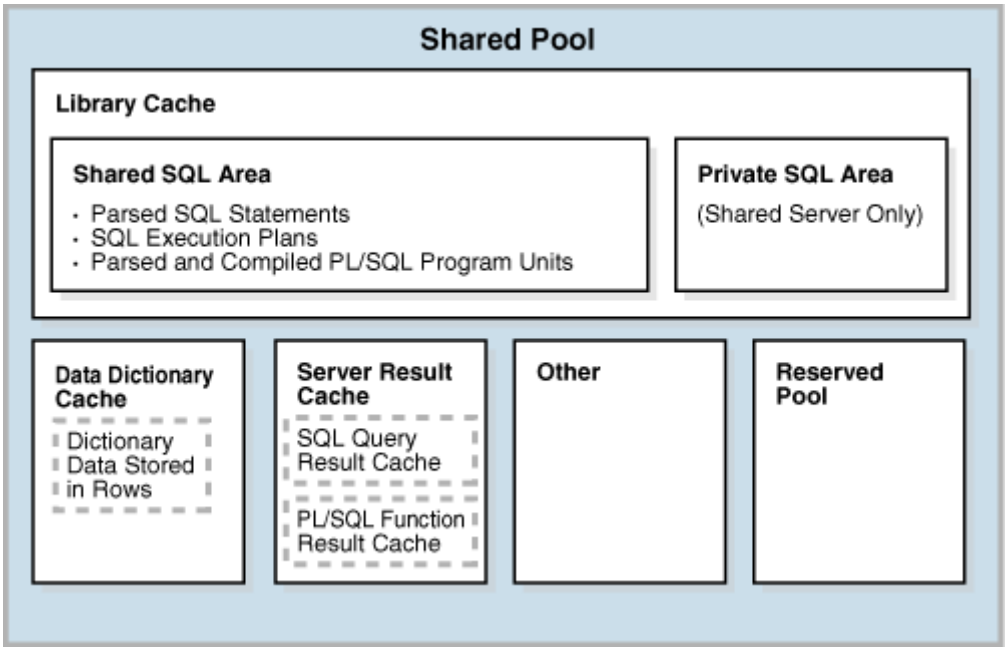
```
create or replace trigger sys.trg_logon
after logon on database
declare
begin
  ...
exception
  when others then
     NULL;
end trg_logon;
```

2.2 Sledování spuštěných dotazů

Sledování dotazů, které uživatelé spouští je poměrně složité. Pokud se spojíme pouze s některými dotazy, zejména těmi, které jsou významné po výkonnostní stránce (jejích zpracování trvá dlouho), můžeme data čerpat z aktuální Library Cache (viz kapitola Library Cache), nebo z AWR reportů (viz kapitola AWR Reporty). Pokud ovšem chceme vědět o každém dotazu, který uživatel spouští, musíme si aktivovat FGA (Viz FGA kapitola).

2.2.1 Library Cache

Je vyrovnávací paměť pro dotazy, která se nachází ve sdílené části paměti instance v SGA, konkrétní části shared pool (umístění viz Obrázek 1). Library cache se obnovuje každých 5 vteřin a obsahuje spouštěné dotazy včetně jejich exekučních plánů. Primární úloha LC je právě uchování exekučních plánů, aby se pro opakující dotazy nemusely stále znovu vytvářet. LC pracuje s LRU policy (10).



Obrázek 2 – Umístění library cache v paměťových strukturách
(Zdroj: http://docs.oracle.com/cd/E11882_01/server.112/e40540/img/cncpt225.gif)

2.2.2 AWR

Pod zkratkou AWR se skrývá užitečný nástroj zvaný Automatic Workload Repository. Tento nástroj patří mezi ostatní, které umožňují hlídat a vyhodnocovat chování a vytížení databázového serveru Oracle. Sběr a analýza dat probíhá tak, že v definovaných intervalech je vytvořen takzvaný snapshot (snímek), který obsahuje informace o aktuálním vytížení serveru, a také se do něho dostanou dotazy, které významně ovlivnily chod serveru (jejich zpracování trvalo dlouho) (11).

K těmto datům je možné přistupovat přes pohledy datového slovníku, nebo přes AWR reporty, které pak zobrazují sumarizovaná data. AWR report je dostupný z aplikace EM, která slouží pro správu databáze. Pro naši potřebu si však vystačíme pouze s daty, která získáme z pohledů datového slovníku. Mezi pohledy AWR mimo jiné patří (11):

- `V$ACTIVE_SESSION_HISTORY` – Zobrazuje aktuální informace o session (ASH), které jsou vzorkovány každou vteřinu.
- `DBA_HIST_ACTIVE_SESS_HISTORY` – Zobrazuje historii pro ASH.
- `DBA_HIST_SNAPSHOT` – Zobrazuje informace z jednotlivých snímků.
- `DBA_HIST_SQL_PLAN` – Zobrazuje historii exekučních plánů.
- `DBA_HIST_WR_CONTROL` – Zobrazuje nastavení AWR.

2.2.3 FGA

Fine Grained Audition (FGA) přinesla od verze 9i možnost, aby si uživatelé sami vytvářeli pravidla pro sběr informací. Doslovný překlad názvu, jemno-zrnný audit, napovídá, že poskytuje prostor pro opravdu detailní sběr informací (12).

Pro lepší pochopení možností FGA si uvedeme jednoduchý příklad. Máme ve firmě úředníka, který má oprávnění na sledování mezd zaměstnanců, ale pouze těch, kteří dosahují mzdy menší než 100 000Kč. Vytvoříme si tedy jednoduché pravidlo pomocí balíčku DBMS_FGA:

```
DBMS_FGA.add_policy(
  object_schema   => 'SCOTT',
  object_name     => 'EMPLOYEES',
  policy_name     => 'SALARY_CHK_AUDIT',
  audit_condition => 'SALARY > 100000',
  audit_column    => 'SALARY');
```

Od teď, pokud se bude někdo dotazovat na řádek, který obsahuje hodnotu vyšší než 100 000Kč, bude o tom uložený záznam do tabulky FGA_LOG$ (12).

2.3 Sledování změny databázových objektů

Pokud chceme sledovat změny databázových objektů, musíme použít vlastní řešení, protože databázový server si pamatuje pouze poslední stav objektu. Ke sledování nám tedy pomůže trigger, který se bude aktivovat vždy, když dojde k nějaké DDL operaci.

Samotný trigger nám neposkytne zdrojový kód objektu, ale pouze informaci o tom, že u objektu došlo ke změně. Pro získání zdrojového kódu můžeme využít několik způsobů. Prvním je použití balíčku DBMS_METADATA. Další možností je využití přímo pohledů datového slovníku ALL_SOURCE (případně alternativy USER_SOURCE, DBA_SOURCE). Tento pohled ale poskytuje pouze zdrojové kódy procedur, funkcí a balíčků.

2.3.1 DBMS_METADATA

Balíček poskytuje spoustu funkcí a jednou z nich je možnost získání DDL databázových objektů. K tomuto účelu slouží funkce GET_DDL, která má následující syntaxi:

```
DBMS_METADATA.GET_DDL (
    object_type      IN VARCHAR2,
    name             IN VARCHAR2,
    schema           IN VARCHAR2 DEFAULT NULL,
    version          IN VARCHAR2 DEFAULT 'COMPATIBLE',
    model            IN VARCHAR2 DEFAULT 'ORACLE',
    transform        IN VARCHAR2 DEFAULT 'DDL')
RETURN CLOB;
```

Výstupem z funkce je formátovaný zdrojový kód objektu, který je poslaný jako parametr. Tato funkce zobrazí DDL pro všechny typy objektů. Pro používání této funkce je třeba speciálních oprávnění správce (role DBA) (13).

```
CREATE TABLE "ST36063"."TRIGGER_ERROR_LOG"
 (          "ID" NUMBER NOT NULL ENABLE,
            "trigger" VARCHAR2(50) NOT NULL ENABLE,
            "KOD_CHYBY" VARCHAR2(20),
            "SCHEMA" VARCHAR2(200),
            "OBJEKT" VARCHAR2(200),
            "TYP_OBJEKTU" NUMBER NOT NULL ENABLE,
             CONSTRAINT "TRIGGER_ERROR_LOG_PK" PRIMARY KEY ("ID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS NOCOMPRESS LOGGING
TABLESPACE "STUDENTS"  ENABLE,
             CONSTRAINT "TRG_ERR_LOG_TYPY_OBJ_FK" FOREIGN KEY ("TYP_OBJEKTU")
              REFERENCES "ST36063"."TYPY_OBJEKTU" ("ID") ENABLE
 ) SEGMENT CREATION DEFERRED
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
TABLESPACE "STUDENTS"
```

Obrázek 3 - Formátovaný výstup z funkce GET_DDL

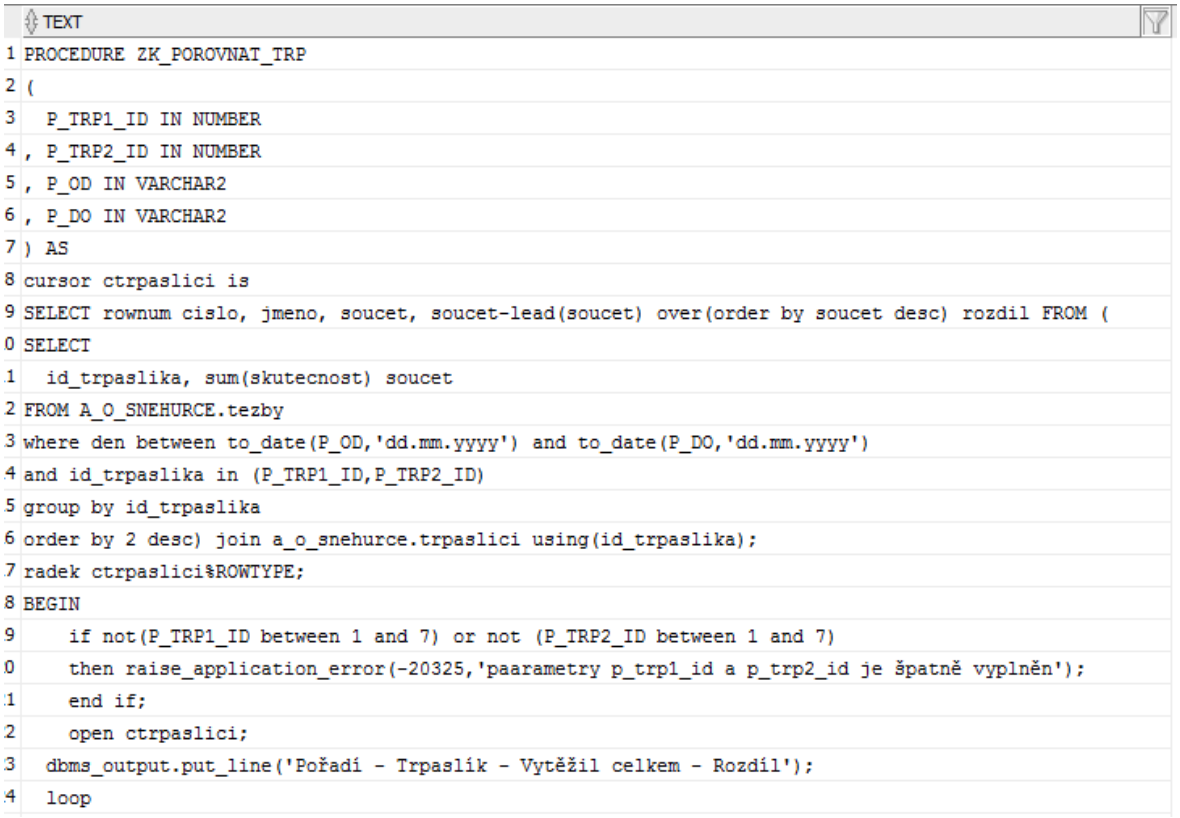
2.3.2 Pohledy datového slovníku

Pohledy poskytují zdrojový kód pouze u funkcí, procedur a balíčků. O to jednodušší je však jeho použití. K dispozici jsou pohledy ALL_SOURCE, USER_SOURCE a DBA_SOURCE. Jednotlivé verze pohledu zobrazují data na základě oprávnění. ALL zobrazuje všechna data (tedy všechny zdrojové kódy objektů), na která má daný uživatel právo. USER zobrazuje zdrojové kódy pouze pro aktuálního uživatele a DBA obsahuje stejný rozsah jako ALL, navíc však poskytuje další podrobnější informace (14).

Zdrojový kód můžeme získat jednoduchým dotazem:

```
select text from dba_source where owner='ST36063' and
name='ZK_POROVNAT_TRP' order by line asc;
```

Výstupem dotazu je zdrojový kód. Každý řádek kódu je vrácen na samostatném řádku výsledku. Část výstupu z dotazu je zobrazen na obrázku (Obrázek 4).



	TEXT
1	PROCEDURE ZK_POROVNAT_TRP
2	(
3	P_TRP1_ID IN NUMBER
4	, P_TRP2_ID IN NUMBER
5	, P_OD IN VARCHAR2
6	, P_DO IN VARCHAR2
7	) AS
8	cursor ctrpaslici is
9	SELECT rownum cislo, jmeno, soucet, soucet-lead(soucet) over(order by soucet desc) rozdil FROM (
0	SELECT
1	id_trpaslika, sum(skutecnost) soucet
2	FROM A_O_SNEHURCE.tezby
3	where den between to_date(P_OD,'dd.mm.yyyy') and to_date(P_DO,'dd.mm.yyyy')
4	and id_trpaslika in (P_TRP1_ID,P_TRP2_ID)
5	group by id_trpaslika
6	order by 2 desc) join a_o_snehurce.trpaslici using(id_trpaslika);
7	radek ctrpaslici%ROWTYPE;
8	BEGIN
9	if not(P_TRP1_ID between 1 and 7) or not (P_TRP2_ID between 1 and 7)
0	then raise_application_error(-20325,'paarametry p_trp1_id a p_trp2_id je špatně vyplněn');
1	end if;
2	open ctrpaslici;
3	dbms_output.put_line('Pořadí - Trpaslík - Vytěžil celkem - Rozdíl');
4	loop

Obrázek 4 - Ukázka výstupu z pohledu DBA_SOURCE

3 Aplikace pro sledování chování uživatelů na databázovém serveru

Aplikace, která byla vytvořena, slouží k získávání a zobrazování informací o činnosti uživatelů v prostředí databázového serveru Oracle 11g2. Aplikace umožňuje zobrazit základní informace o uživateli, jeho napojení na databázové předměty, a také poskytuje přehled dotazů, které uživatel spouštěl. Dotazy jsou k dispozici jednak z aktuální cache a také z AWR reportů.

Uživatelé mohou přistupovat ve 4 režimech. První režim je nepřihlášený uživatel, který nemůže téměř nic, jelikož jde o důvěrná data. Další režim je přihlášení běžného uživatele, který může vidět svá data, pokud je jeho účet spojený s účtem v databázi. Další hladina zabezpečení je správce skupiny, ten může sledovat uživatele, kteří jsou připojeni do jeho skupiny a poslední je administrátor, který má absolutní práva. Přihlášení do aplikace je odstíněno od přihlášení do databáze. Uživatel, který se dostane do databáze, se nemusí automaticky dostat do aplikace.

Aplikace je primárně určena pro výukové účely databázových systémů. Aktuální verze je k dispozici na vnitřní síti univerzity, na adrese http://fei-db-info.upceucebny.cz.
V současné době není aplikace k dispozici pro všechny uživatele - zobrazuje data pouze pro registrované uživatele.

3.1 Návrh a použité technologie

3.1.1 Datová vrstva

Základem databázové vrstvy je jednoduchý model, který umožňuje shromažďovat základní informace pro správný chod aplikace. Model byl vytvořený v programu Oracle SQL Developer Datamodeler.

Aplikace využívá základní performance pohledy z datového slovníku databáze. Dále využívá schéma C_CORE, které umožňuje napojení jednotlivých uživatelů na seznamy studentů jednotlivých databázových předmětů. Je tedy možné spárovat databázové schéma s konkrétním jménem studenta.

Další část informací je realizovaná automatickým sběrem pomocí triggeru. Tento trigger při každé DDL operaci zaznamená informace o provedené operaci. U každého databázového objektu tedy bude sledovaná kompletní historie jednotlivých databázových objektů jednotlivých uživatelů. Trigger je navržen tak, aby nenarušil bezproblémový chod databázového serveru a byl pro uživatele naprosto transparentní.

3.1.2 Tabulky

3.1.2.1 DB_OBJEKTY

Tabulka slouží pro uchovávání informací o aktuální verzi jednotlivých databázových objektů. Cizí klíč na id_uzivatele, umožňuje napojení na konkrétního uživatele aplikace. Tabulka je plněna automaticky pomocí triggeru.

```
CREATE TABLE DB_OBJEKTY
(
schema VARCHAR2 (200) NOT NULL ,
nazev VARCHAR2 (200) NOT NULL ,
typ_objektu NUMBER NOT NULL ,
operace VARCHAR2 (10) NOT NULL ,
casove_razitko TIMESTAMP ,
id_uzivatele NUMBER ,
zdrojovy_kod CLOB
) ;
ALTER TABLE DB_OBJEKTY ADD CONSTRAINT object_log_PK PRIMARY
KEY ( schema, nazev ) ;
ALTER TABLE DB_OBJEKTY ADD CONSTRAINT
DB_OBJEKTY_TYPY_OBJEKTU_FK FOREIGN KEY ( typ_objektu )
REFERENCES TYPY_OBJEKTU ( ID ) ;
ALTER TABLE DB_OBJEKTY ADD CONSTRAINT DB_OBJEKTY_UZIVATELE_FK
FOREIGN KEY ( id_uzivatele ) REFERENCES UZIVATELE ( id ) ;
```

3.1.2.2 DB_OBJEKTY_HISTORIE

Tabulka slouží k uchovávání historických informací o databázových objektech. Tato tabulka je plněna automaticky pomocí triggeru v okamžiku, kdy je upravovaný záznam v tabulce DB_OBJEKTY.

```
CREATE TABLE DB_OBJEKTY_HISTORIE
(
id NUMBER NOT NULL ,
schema VARCHAR2 (200) NOT NULL ,
nazev VARCHAR2 (200) NOT NULL ,
typ_objektu NUMBER NOT NULL ,
casove_razitko TIMESTAMP NOT NULL ,
zdrojovy_kod CLOB
) ;
ALTER TABLE DB_OBJEKTY_HISTORIE ADD CONSTRAINT
DB_OBJEKTY_HISTORIE_PK PRIMARY KEY ( id ) ;
ALTER TABLE DB_OBJEKTY_HISTORIE ADD CONSTRAINT
DB_OBJEKTY_HIST_DB_OBJ_FK FOREIGN KEY ( schema, nazev )
REFERENCES DB_OBJEKTY ( schema, nazev ) ;
ALTER TABLE DB_OBJEKTY_HISTORIE ADD CONSTRAINT
DB_OBJEKTY_HIST_TYPY_OBJ_FK FOREIGN KEY ( typ_objektu )
REFERENCES TYPY_OBJEKTU ( ID ) ;
```

Primární klíč tabulky id je generován automaticky pomocí sekvence a plněno pomocí triggeru. Zdrojový kód triggeru následuje v další ukázce.

```
CREATE SEQUENCE DB_OBJEKTY_HISTORIE_ID_SEQ START WITH 1
NOCACHE ORDER ;
CREATE OR REPLACE TRIGGER DB_OBJEKTY_HISTORIE_id_TRG BEFORE
INSERT ON DB_OBJEKTY_HISTORIE FOR EACH ROW WHEN (NEW.id IS
NULL)
BEGIN
     :NEW.id := DB_OBJEKTY_HISTORIE_ID_SEQ.NEXTVAL;
END;
```

3.1.2.3 ROLE

Číselníková tabulka obsahuje definice rolí pro uživatele aplikace. Aplikace neprovádí žádné změny v datech této tabulky.

```
CREATE TABLE ROLE
(
    kod CHAR (2) NOT NULL ,
    nazev VARCHAR2 (20) NOT NULL
) ;
ALTER TABLE ROLE ADD CONSTRAINT ROLE_PK PRIMARY KEY ( kod ) ;
```

3.1.2.4 TYPY_OBJEKTU

Číselníková tabulka pro jednotlivé typy objektů.

```
CREATE TABLE TYPY_OBJEKTU
(
ID NUMBER NOT NULL ,
Oznaceni VARCHAR2 (200) NOT NULL
) ;
ALTER TABLE TYPY_OBJEKTU ADD CONSTRAINT TYPY_OBJEKTU_PK
PRIMARY KEY ( ID ) ;
```

Id je generováno automaticky pomocí sekvence a plněno pomocí triggeru.

```
CREATE SEQUENCE TYPY_OBJEKTU_ID_SEQ START WITH 1 NOCACHE
ORDER ;
CREATE OR REPLACE TRIGGER TYPY_OBJEKTU_ID_TRG BEFORE INSERT
ON TYPY_OBJEKTU FOR EACH ROW WHEN (NEW.id IS NULL)
BEGIN
     :NEW.id := TYPY_OBJEKTU_ID_SEQ.NEXTVAL;
END;
```

3.1.3 Systémové pohledy

Velké množství informací si aplikace nemusí generovat sama, ale stačí využít existujících systémových pohledů, které již informace poskytují. V této kapitole jsou uvedeny a popsány základní systémové pohledy, které jsou použity v aplikaci.

3.1.3.1 V$SQL

Pohled V$SQL zobrazuje statistiky ze sdílené SQL oblasti (Library Cache) a zobrazuje vždy jeden řádek pro jedno unikátní SQL. Zobrazované statistiky jsou aktualizovány vždy po dokončení exekuce dotazu. Pokud je exekuce dotazu dlouhá, je prováděn update statistik každých 5 s. (15)

Tabulka 1 - Vybrané sloupce z pohled V$SQL

Sloupec	Datový typ	Význam
EXECUTIONS	NUMBER	Počet spuštění dotazu od doby, co byl zařazen do library cache
PARSING_SCHEMA_NAME	VARCHAR2(30)	Schéma, ve kterém byl dotaz poprvé parsován
SERVICE	VARCHAR2(64)	Název služby
MODULE	VARCHAR2(64)	Název modulu, který byl použit při prvním spuštění dotazu.
ACTION	VARCHAR2(64)	Název akce, který byl použit při prvním spuštění dotazu.
LAST_ACTIVE_TIME	DATE	Datum a čas, kdy byl plán dotazu naposledy aktivní.
ROWS_PROCESSED	NUMBER	Celkový počet řádků, které dotaz vrátil.
CHILD_NUMBER	NUMBER	Počet child kurzorů (alternativních plánů)
SQL_ID	VARCHAR2(13)	SQL identifikátor kurzoru

3.1.3.2 V$SESSION

Druhý z takzvaných performance view, tedy pohled, který zobrazuje aktuální aktivní data, je V$SESSION. Pohled poskytuje informace o aktuálním seznamu aktivních session, který uživatel je přihlášen a ze které stanice. Dále také tento pohled poskytuje informace o akci, kterou session právě provádí. Případně o akci, na kterou session právě čeká. V tabulce níže jsou zobrazeny sloupce, které jsou použity v práci. (16)

Tabulka 2 - Vybrané sloupce z pohledu V$SESSION

Sloupec	Datový typ	Význam
SID	NUMBER	Identifikátor session.
SERIAL#	NUMBER	Unikátní identifikátor objektů patřících session. Zajišťuje unikátnost, pokud je SID využito další session.
USERNAME	VARCHAR2(30)	Oracle uživatelské jméno.
STATUS	VARCHAR2(8)	Aktuální status.
MACHINE	VARCHAR2(64)	Název počítače.
PROGRAM	VARCHAR2(48)	Název programu.
EVENT	VARCHAR2(64)	Událost, na které aktuálně session tráví čas.

3.1.3.3 DBA_AUDIT_TRAIL

Zobrazuje informace z auditu databáze. Tento pohled poskytuje informace o akcích, jako jsou přihlášení a odhlášení uživatelů. Mimo jiné poskytuje informace o právech a rolích, se kterými se uživatel přihlašuje. V tabulce níže je zobrazený seznam sloupců, které jsou relevantní pro tuto práci. (9)

Tabulka 3 - Vybrané sloupce z pohledu DBA_AUDIT_TRAIL

Sloupec	Datový typ	Význam
OS_USERNAME	VARCHAR2(255)	Uživatelské jméno v systému, který provádí akci, která je auditována.
USERHOST	VARCHAR2(128)	Název klientské stanice.
USERNAME	VARCHAR2(30)	Název db uživatele.
TIMESTAMP	DATE	Datum a čas vzniku události.
ACTION_NAME	VARCHAR2(28)	Název akce, která byla provedena.

3.1.4 Ostatní databázové objekty

3.1.4.1 Funkce RAW_TO_DATE

Funkce akceptuje vstupní parametr typu RAW a provádí jeho převod do formátu DATE.

```
create or replace function raw_to_date(i_raw RAW) return date
as
    i_date DATE DEFAULT NULL;
begin
    dbms_stats.convert_raw_value(i_raw, i_date);
    return i_date;
end raw_to_date;
```

3.1.4.2 Funkce jeAktivniUzivatel

Funkce na základě parametru, který akceptuje název uživatele, zjistí, zda se uživatel přihlásil k databázi během posledních 120dní. Pokud ano, je uživatel vyhodnocen jako aktivní a je vrácena hodnota 1.

```
create or replace function jeAktivniUzivatel(iUser IN
VARCHAR2)return number
as
    iCnt NUMBER;
    cursor iCrs is SELECT /*+ PARALLEL(16) */ 1 FROM
    dba_audit_trail
    WHERE username=iUser and timestamp > sysdate-120 and
    action_name='LOGON' and rownum = 1;
begin
    iCnt := 0;
    open iCrs;
        fetch iCrs into iCnt;
    close iCrs;
    return iCnt;
end jeAktivniUzivatel;
```

3.1.4.3 Trigger trg_ddl_log

Trigger slouží pro získávání informací o vytvářených a měněných databázových objektech.

```
CREATE OR REPLACE TRIGGER trg_ddl_log AFTER DDL ON DATABASE
DECLARE
  i_error VARCHAR2(50);
  i_errm CLOB;
  i_job_name VARCHAR2(100);
  i_ddl CLOB;
BEGIN
    BEGIN
      dbms_lob.CREATETEMPORARY(i_ddl, FALSE);
      i_ddl :=
SYS.dbcheck_get_dll(UPPER(ora_dict_obj_owner),UPPER(ora_dict_
obj_name),UPPER(ora_dict_obj_type));
    EXCEPTION
      WHEN OTHERS THEN
        i_ddl := SQLERRM;
    END;
    BEGIN
      insert into C_DB_INFO.DB_OBJEKTY(SCHEMA, nazev,
typ_objektu, casove_razitko, zdrojovy_kod, operace)
      values(
        upper(ora_dict_obj_owner),
        upper(ora_dict_obj_name),
        (select id from c_db_info.typy_objektu where
oznaceni=ora_dict_obj_type),
        systimestamp,
        i_ddl,
        0
      );
    EXCEPTION
      WHEN OTHERS THEN
        NULL;
    END;
EXCEPTION
  WHEN OTHERS THEN
    NULL;
END trg_ddl_log;
```

3.1.4.4 DB_OBJEKTY_HIST_TRG

Trigger, který slouží pro zachování historie databázových objektů. Aby bylo možné v triggeru provádět operaci commit, je nutné, aby trigger běžel v režimu autonomní transakce. Ta funguje tak, že aktuálně běžící transakce je pozastavena, provedou se všechny operace v rámci autonomní transakce a následně je původní transakce opět obnovena. Úspěch nebo neúspěch autonomní transakce nemá žádný vliv na původní transakci a naopak. Standardně se využívá právě pro logování operací, kdy potřebujeme mít v logu zapsané informace i v případě, že hlavní transakce skončí rollbackem.

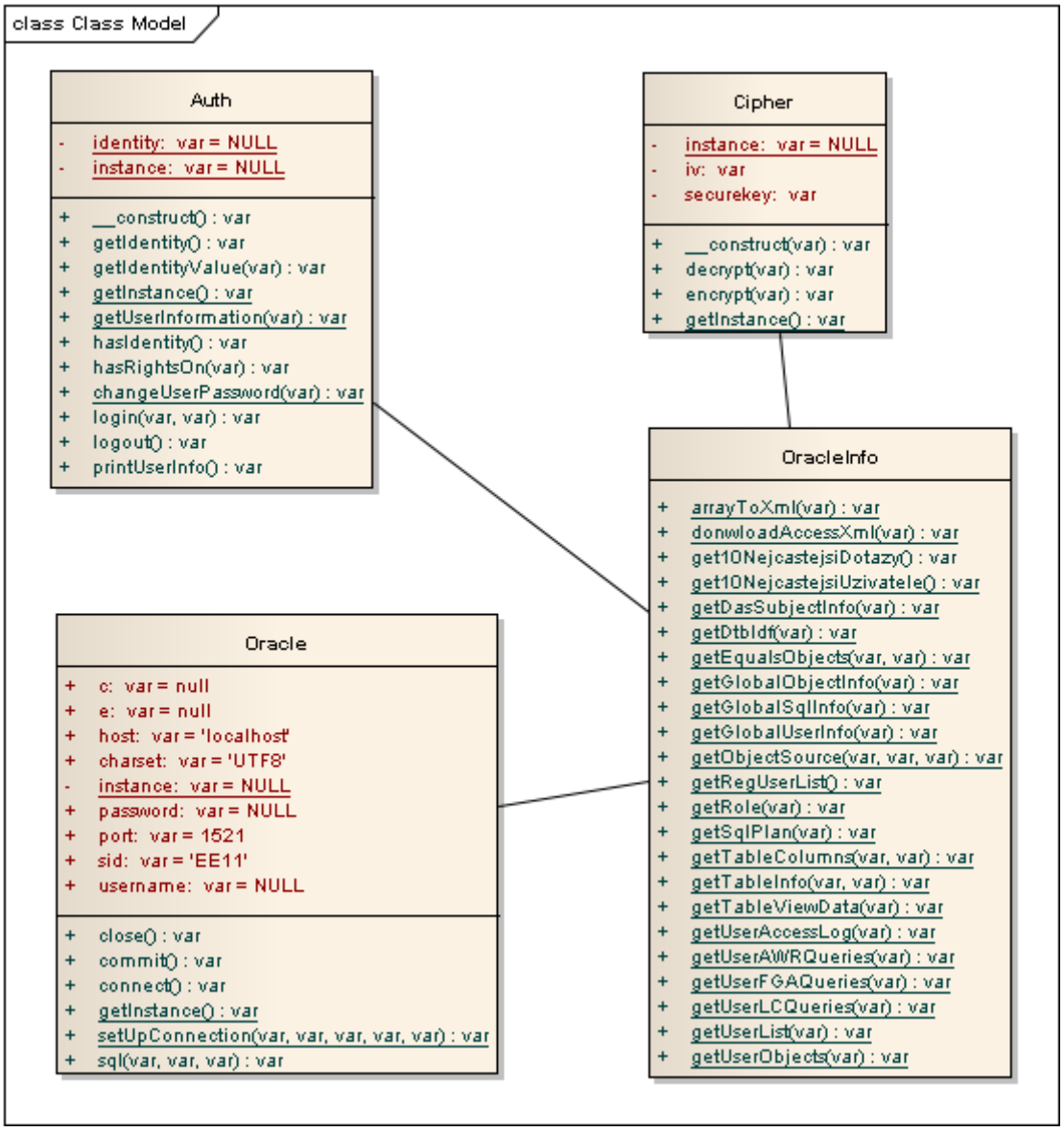
```
CREATE OR REPLACE TRIGGER C_DB_INFO.DB_OBJEKTY_HIST_TRG
  BEFORE INSERT ON C_DB_INFO.DB_OBJEKTY FOR EACH ROW
DECLARE
  PRAGMA AUTONOMOUS_TRANSACTION;
  i_cnt NUMBER;
BEGIN
  NULL;
  SELECT COUNT(*) INTO i_cnt FROM db_objekty WHERE
SCHEMA=:new.SCHEMA AND nazev=:new.nazev;
  IF i_cnt > 0 THEN
    INSERT INTO db_objekty_historie(SCHEMA, nazev,
typ_objektu, casove_razitko, zdrojovy_kod)
      SELECT SCHEMA, nazev, typ_objektu, casove_razitko,
zdrojovy_kod FROM db_objekty WHERE
        SCHEMA=:new.SCHEMA AND nazev=:new.nazev;
    UPDATE db_objekty SET casove_razitko=:new.casove_razitko,
zdrojovy_kod=:new.zdrojovy_kod WHERE
    SCHEMA=:new.SCHEMA AND nazev=:new.nazev;
    COMMIT;
  END IF;
END;
```

3.2 Architektura aplikace

Aplikace je zaměřena na poskytování relevantních informací a její zpracování, je tedy jednoduššího rázu. Důraz nebyl kladen ani na grafickou stránku aplikace. Při vývoji byl použit jazyk PHP spolu s HTML5.

Jádro aplikace je tvořeno třemi třídami, které se starají o veškerou funkcionalitu. Třídy jsou uvedeny v diagramu tříd (Obrázek 5). Třída Auth se stará o zabezpečení aplikace. Přes tuto třídu probíhá přihlašování uživatelů a třída také udržuje informace o identitě přihlášeného uživatele. Metoda přihlášení je zobrazena v aktivity digramu (Obrázek 7).

Třída Oracle slouží k navázání připojení k databázovému serveru. Přes metodu setUpConnection je nastaven připojovací řetězec a je vytvořena instance připojení. Při použití je následně vyžádána instance připojení a po provedení dotazu je toto připojení ukončeno.



Obrázek 5 - Diagram tříd

3.2.1 Třída Cipher

Jak napovídá název třídy, Cipher slouží k šifrování. Tato třída je využívána pro šifrování požadavků, které se předávají v adresním řádku (metoda GET). Cipher je opět připravena podle návrhového vzoru singleton. Při inicializaci aplikace je vygenerovaný klíč, kterým se potom po celou dobu požadavky šifrují.

3.2.2 Třída Oracle

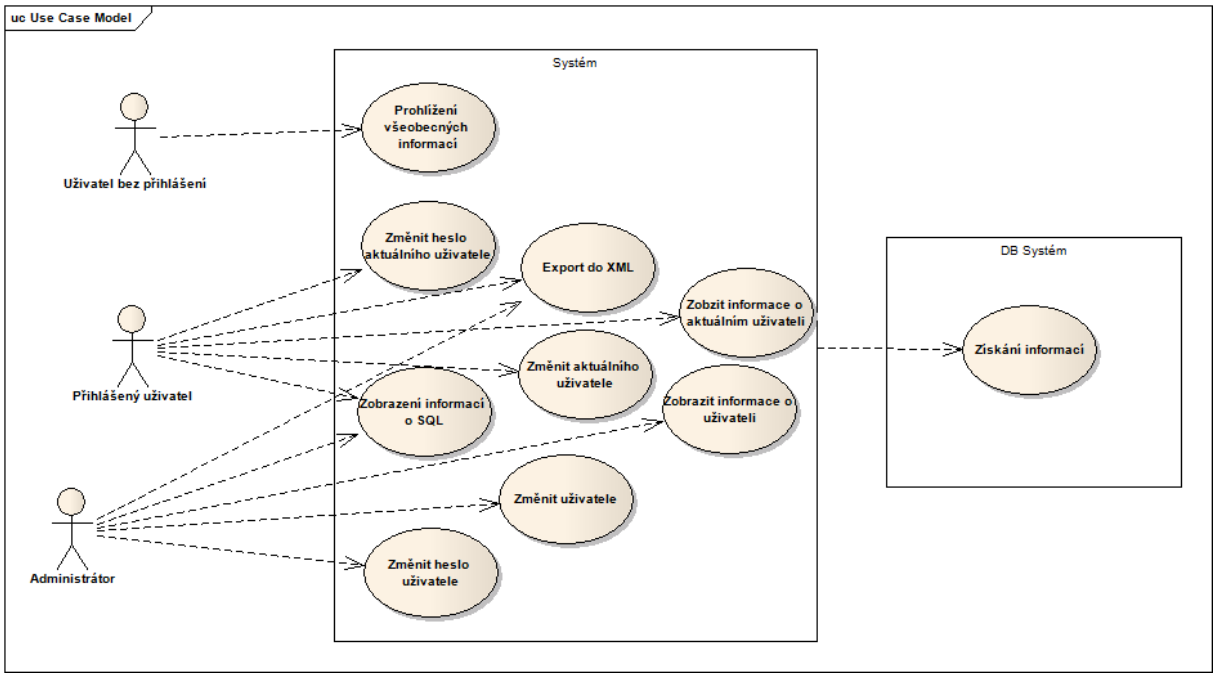
Tato třída slouží k navázání spojení s databázovým serverem. Psaná je dle návrhového vzoru sigleton. Hlavní metodou je setUpConnection, která naváže spojení se serverem. Dále je potom připojení realizováno pomocí jedné instance, která je ve třídě připravená a dostupná přes metodu getInstance().

3.2.3 Třída Auth

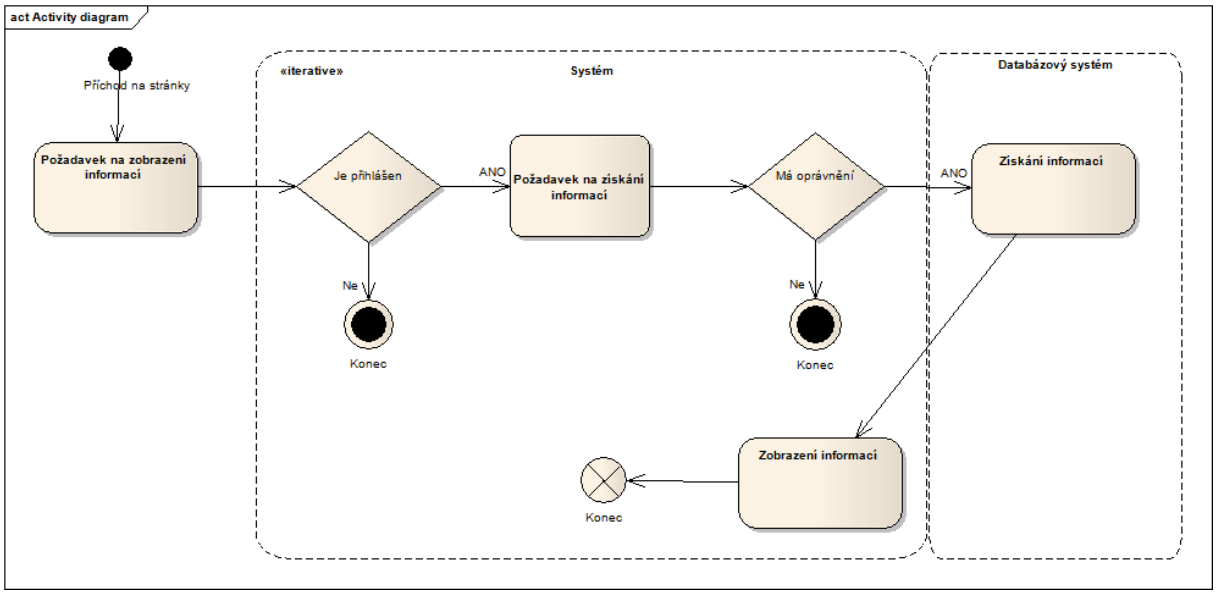
Nejdůležitější třída z pohledu zabezpečení aplikace je třída Auth. V této třídě je udržována identita přihlášeného uživatele. Třída má také na starost vyhodnocovat oprávnění uživatele na jednotlivé aplikační moduly. Implementace třídy je také provedena podle návrhového vzoru singleton. Instance třídy je k dispozici po volání metody getInstance().

3.2.4 Třída OracleInfo

Třída OracleInfo je hlavní třídou celé aplikace. Tato třída pomocí statických metod poskytuje veškeré informace, které aplikace zobrazuje. V této kapitole si povíme více o jednotlivých metodách a o informacích, které poskytují. Základní sada funkcí, které jsou poskytovány touto třídou, jsou v závislosti na uživatelských rolích zobrazeny v use-case diagramu (Obrázek 6).

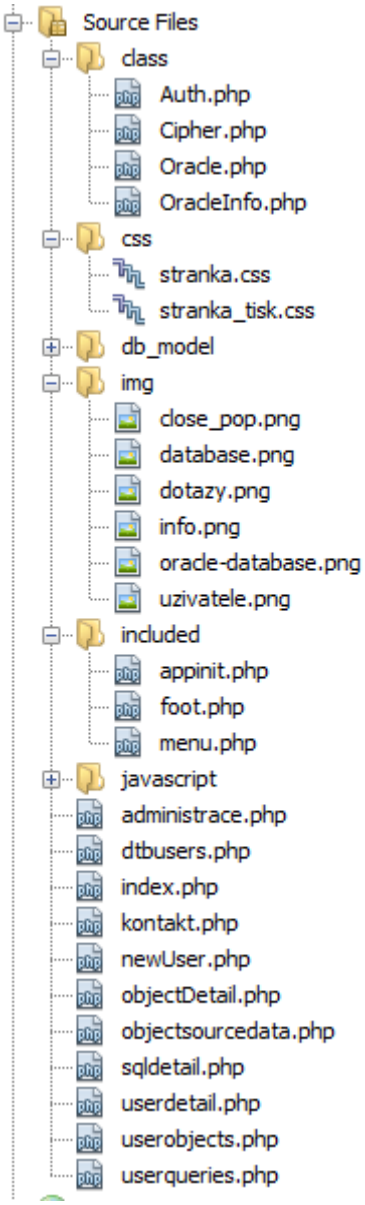


Obrázek 6 - Use case diagram



Obrázek 7- Activity diagram

3.3 Adresářová struktura



Základem aplikace jsou soubory v adresáři class, které obsahují základní třídy, konkrétně třídy Auth, která slouží ke správě přístupu k aplikaci, dále je zde třída Oracle, která využívá návrhového vzoru singleton a je určena k řízení přístupu k databázi. Poslední třídou je třída OracleInfo, která pomocí statických metod poskytuje kompletní informace zobrazené v aplikaci.

Adresář CSS obsahuje soubory s kaskádovými styly. Soubor stranka.css je určen pro normální zobrazení stránek, soubor stranka_tisk.css obsahuje styly optimalizované pro tiskárny.

Adresář img obsahuje veškerou grafiku, která se v aplikaci objevuje.

V adresáři included jsou společné části pro všechny podstránky. Je zde navigace, inicializace a patička aplikace. Společné části jsou vkládány pomocí funkce include.

Adresář javascript obsahuje základní knihovnu jQuery, kterou aplikace využívá pro zobrazení popup oken.

V hlavním adresáři aplikace jsou jednotlivé stránky aplikace. Základní stránkou je soubor index.php.

3.4 Popis fungování aplikace

Základní navigace aplikace je rozdělena na:

- Úvod.
- Informace.
- Administrace.
- Kontakt.
- Přihlášení.

3.4.1 Úvod

Úvod aplikace je definovaný v souboru index.php. Slouží jako rozcestník celé aplikace. Stránka zobrazuje informace o 10 aktuálních dotazech a počtu jejich spuštění. Pokud je na této stránce uživatel přihlášený jako administrátor, má k dispozici po kliknutí na konkrétní dotaz detail toho dotazu. V detailu je k dispozici základní statistika a také aktuální exekuční plán.

3.4.2 Informace

V modulu informace jsou k dispozici tabulky jednotlivých uživatelů. Množství dat, které se v tabulce zobrazuje, je odvozeno od role, kterou má aktuálně přihlášený uživatel přidělenou. Pokud uživatel není přihlášený, nejsou data k dispozici vůbec, pokud je uživatel přihlášený v roli uživatel, má k dispozici pouze své vlastní informace. Uživatel s rolí správce skupiny má k dispozici informace o uživatelích, kteří jsou přiřazeni do jeho skupiny. Administrátor má k dispozici veškeré informace o všech uživatelích, kteří jsou v databázi.

V zobrazeném seznamu je k dispozici odkaz na detail uživatele (kapitola 3.4.6) a také vyhledávací pole, které umožňuje přímý přechod na konkrétního uživatele. Přechod je možný i na uživatele, který se standardně v seznamu nezobrazuje. K těmto uživatelům patří testovací schémata, schémata vyučujících atp.

Obrázek 8 - Uživatelé databázového serveru

3.4.3 Administrace

V části administrace je k dispozici seznam registrovaných uživatelů v systému. Uživatelé systému jsou odstínění od uživatelů, kteří jsou v databázi, a to z bezpečnostních důvodů. Poskytnutý seznam je také odvozený od role, kterou má přiřazenou aktuální uživatel.

Všechny role kromě administrátora mají přístup pouze k vlastnímu účtu a nemohou vytvářet nové uživatelské účty.

3.4.4 Kontakt

Zobrazuje informace o autorovi aplikace.

3.4.5 Přihlásit se

Tato položka je zobrazena pouze, pokud není přihlášený žádný uživatel a slouží k zobrazení plovoucího okna pro přihlášení. Jakmile je uživatel přihlášený, je položka navigace změněna na možnost odhlásit se.

3.4.6 Detail uživatele

Detail uživatele je k dispozici z úvodní strany, z přehledu, následně z modulu informace, případně ze seznamu uživatelů, z modulu administrace. V detailu jsou k dispozici základní informace (Obrázek 12), dále napojení na databázové předměty.

Pro zobrazení detailních informací slouží odkazy na stránce. Z detailu je možné přejít na informace z Library Cache (Obrázek 9), které obsahují aktuální seznam dotazů přístupných z cache. Další odkaz směřuje na dotazy z AWR reportů (Obrázek 10), který poskytuje dotazy, které nějakým způsobem ovlivnil chod serveru (například, že dotaz běžel nezanedbatelně dlouho). Třetí odkaz v řadě poskytuje informace z FGA (Obrázek 11). Tento přehled poskytuje veškeré dotazy, které uživatel provedl nad objekty, které jsou zahrnuty do FGA auditu. Poslední odkaz poskytuje přehled všech objektů, které jsou ve schématu uživatele k dispozici.

Aktuálně v LC

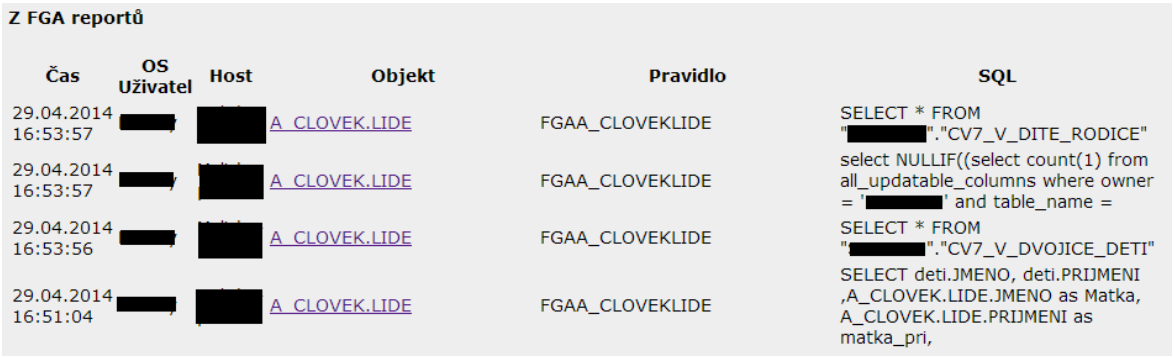
SQL ID	Text
gypnfv5nzurb0	select child_number from v$sql where sql_id = :1 order by child_number

Obrázek 9 - Přehled dotazů z LC

Z AWR reportů

SQL ID	Text
g02nyp5t17vac	/*+ NO_SQL_TRANSLATION */SELECT 'COLUMN' type, owner, table_name object_name, column_name, column_id
a5jphv11dnpk8	/*+ NO_SQL_TRANSLATION */select table_owner, table_name from all_synonyms where owner = 'A_O_SNEHUR

Obrázek 10 - Přehled dotazů z AWR



Z FGA reportů

Čas	OS Uživatel	Host	Objekt	Pravidlo	SQL
29.04.2014 16:53:57	[redacted]	[redacted]	A_CLOVEK.LIDE	FGAA_CLOVEKLIDE	SELECT * FROM "[redacted]"."CV7_V_DITE_RODICE"
29.04.2014 16:53:57	[redacted]	[redacted]	A_CLOVEK.LIDE	FGAA_CLOVEKLIDE	select NULLIF((select count(1) from all_updatable_columns where owner = '[redacted]' and table_name =
29.04.2014 16:53:56	[redacted]	[redacted]	A_CLOVEK.LIDE	FGAA_CLOVEKLIDE	SELECT * FROM "[redacted]"."CV7_V_DVOJICE_DETI"
29.04.2014 16:51:04	[redacted]	[redacted]	A_CLOVEK.LIDE	FGAA_CLOVEKLIDE	SELECT deti.JMENO, deti.PRIJMENI ,A_CLOVEK.LIDE.JMENO as Matka, A_CLOVEK.LIDE.PRIJMENI as matka_pri,

Obrázek 11 - Přehled dotazů z FGA

UŽIVATEL ST36063

Obecné informace

Jméno: **Helena**
Příjmení: **Zechmeisterová**
E-mail: **st36063@student.upce.cz**
Poslední úprava údajů: **01.01.2014 22:42:06**
Schéma: **ST36063**
Vytvoření schématu: **22.02.2013 11:32:53**
Aktivní db uživatel: **Ano**

Informace o vazbě na databázové předměty

A/R	Předmět	Název	Role
2013/2014	KIT/IDAS2	Databázové systémy II	Student
2012/2013	KIT/IDAS1	Databázové systémy I	Student

Uživatelské dotazy z LC Uživatelské dotazy z AWR Uživatelské dotazy z FGA Databázové objekty

Přístupy k databázovému systému

Datum a čas	Operace	Počítač	Uživateslký účet
14.04.2014 09:15:29	LOGOFF	fei-hosting.upceucebny.cz	apache
14.04.2014 09:15:26	LOGON	fei-hosting.upceucebny.cz	apache
14.04.2014 09:15:26	LOGON	fei-hosting.upceucebny.cz	apache
14.04.2014 09:15:26	LOGON	fei-hosting.upceucebny.cz	apache
14.04.2014 09:15:26	LOGON	fei-hosting.upceucebny.cz	apache
14.04.2014 09:15:26	LOGOFF	fei-hosting.upceucebny.cz	apache
14.04.2014 09:15:26	LOGOFF	fei-hosting.upceucebny.cz	apache
14.04.2014 09:15:26	LOGOFF	fei-hosting.upceucebny.cz	apache
14.04.2014 09:15:26	LOGOFF	fei-hosting.upceucebny.cz	apache
14.04.2014 09:15:23	LOGON	fei-hosting.upceucebny.cz	apache

Exportovat všechny záznamy do XML

Obrázek 12 - Přehled informací v detailu uživatele

3.4.7 Detail dotazu

Detail dotazu poskytuje základní charakteristiky o dotazu včetně exekučního plánu. Detail dotazu je k dispozici z úvodní stránky, případně z detailu uživatele. V současné verzi aplikace je detail k dispozici pouze pro dotazy, které se aktuálně nachází v library cache. V této oblasti je tedy prostor pro další rozšíření.

3.4.8 Detail objektu

V modulu databázových objektů je k dispozici přehled všech objektů, které má uživatel ve svém schématu (Obrázek 13).

Databázové objekty

Název	Typ	Vytvořeno	Změněno	Stav
V_MOJE_POKUS		07.04.2014 08:48:50	29.04.2014 15:24:56	VALID
V_MUJ_POKUS		07.04.2014 15:18:45	07.04.2014 15:18:53	VALID
PRODEJE		29.04.2014 09:14:12	29.04.2014 09:14:12	VALID

Obrázek 13 - Přehled databázových objektů

Z tohoto přehledu je možné přejít na detail databázového objektu (Obrázek 14). Detail poskytuje komplexní informace o zvoleném databázovém objektu. Přehled obsahuje informace společné pro všechny typy objektů, jako jsou:

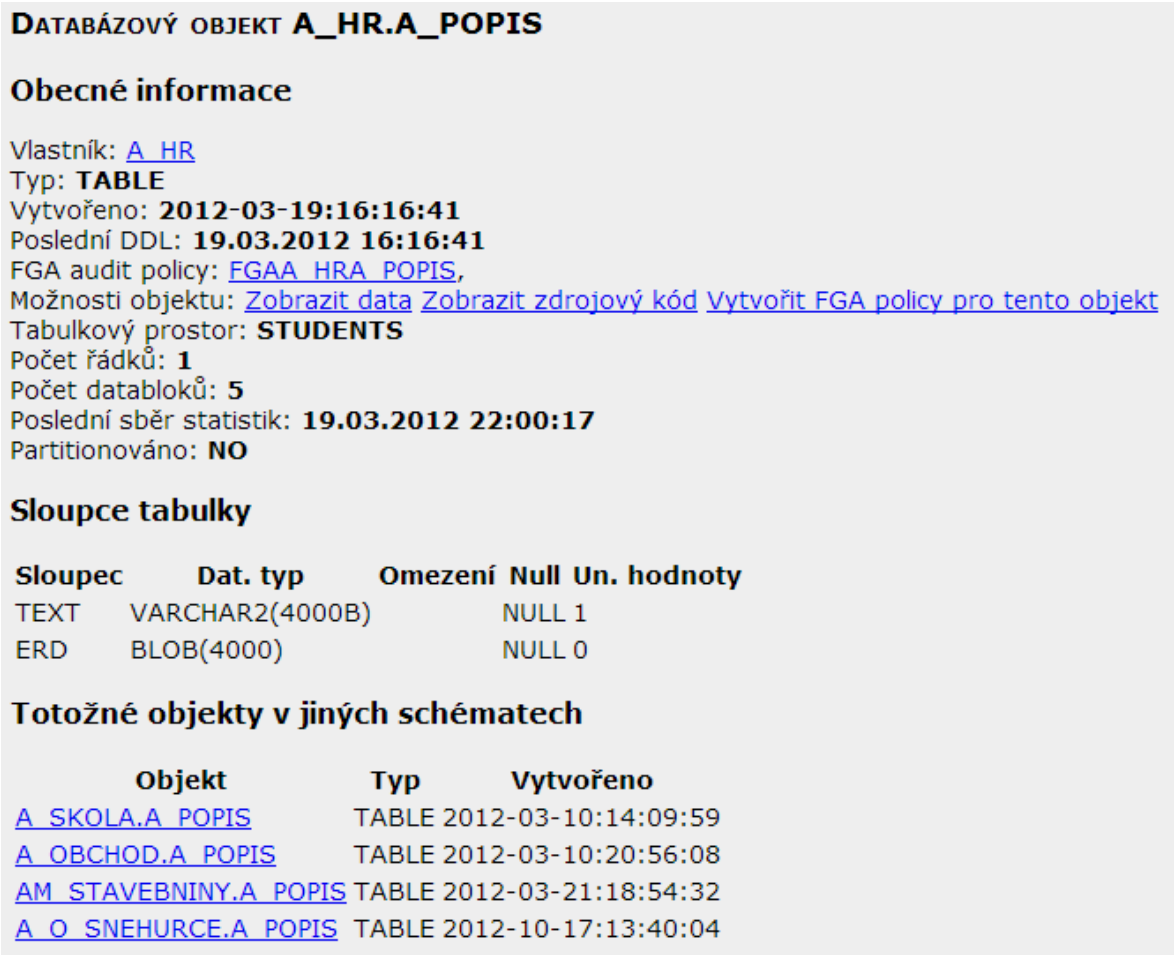
- vlastník,
- typ,
- datum vytvoření,
- datum poslední změny.

Další informace jsou k dispozici pro jednotlivé typy databázových objektů. Nejvíce informací poskytuje aplikace pro databázové objekty pohled a tabulka. Jsou to informace, které obsahují:

- počet řádků,
- počet datových bloků,
- datum a čas posledního sběru statistik,
- tabulkový prostor.

Pro tabulky a pohledy je také k dispozici seznam sloupců spolu s informacemi o datovém typu a omezení, které je na sloupec vázáno. Také je k dispozici odkaz na zobrazení dat objektu.

Poslední sada informací, která je již společná pro všechny typy objektů, je seznam objektů, které mají totožný název v jiných schématech databáze. Dále jsou k dispozici možnosti pro zobrazení zdrojového kódu objektu.



DATABÁZOVÝ OBJEKT A_HR.A_POPIS

Obecné informace

Vlastník: A_HR
Typ: **TABLE**
Vytvořeno: **2012-03-19:16:16:41**
Poslední DDL: **19.03.2012 16:16:41**
FGA audit policy: FGAA_HRA_POPIS,
Možnosti objektu: Zobrazit data Zobrazit zdrojový kód Vytvořit FGA policy pro tento objekt
Tabulkový prostor: **STUDENTS**
Počet řádků: **1**
Počet databloků: **5**
Poslední sběr statistik: **19.03.2012 22:00:17**
Partitionováno: **NO**

Sloupce tabulky

Sloupec	Dat. typ	Omezení	Null	Un. hodnoty
TEXT	VARCHAR2(4000B)		NULL	1
ERD	BLOB(4000)		NULL	0

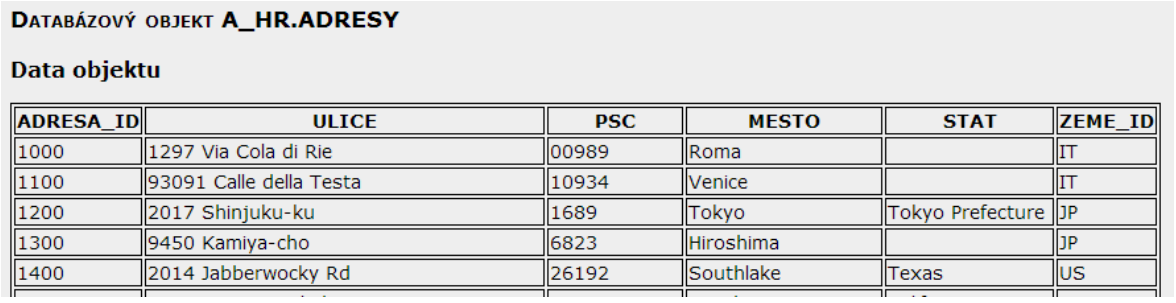
Totožné objekty v jiných schématech

Objekt	Typ	Vytvořeno
A_SKOLA.A_POPIS	TABLE	2012-03-10:14:09:59
A_OBCHOD.A_POPIS	TABLE	2012-03-10:20:56:08
AM_STAVEBNINY.A_POPIS	TABLE	2012-03-21:18:54:32
A_O_SNEHURCE.A_POPIS	TABLE	2012-10-17:13:40:04

Obrázek 14 - Detail databázového objektu

3.4.9 Data databázového objektu

Jak bylo zmíněno již výše, jsou data k dispozici pouze pro objekty typu tabulka a pohled. Jiné objekty totiž neposkytují data standardním způsobem. Výpis data probíhá tak, že nejdříve jsou načteny všechny sloupce objektu a následně je sestaven dynamický dotaz, který načte obsah pro vybrané sloupce (Obrázek 15).



DATABÁZOVÝ OBJEKT A_HR.ADRESY

Data objektu

ADRESA_ID	ULICE	PSC	MESTO	STAT	ZEME_ID
1000	1297 Via Cola di Rie	00989	Roma		IT
1100	93091 Calle della Testa	10934	Venice		IT
1200	2017 Shinjuku-ku	1689	Tokyo	Tokyo Prefecture	JP
1300	9450 Kamiya-cho	6823	Hiroshima		JP
1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US

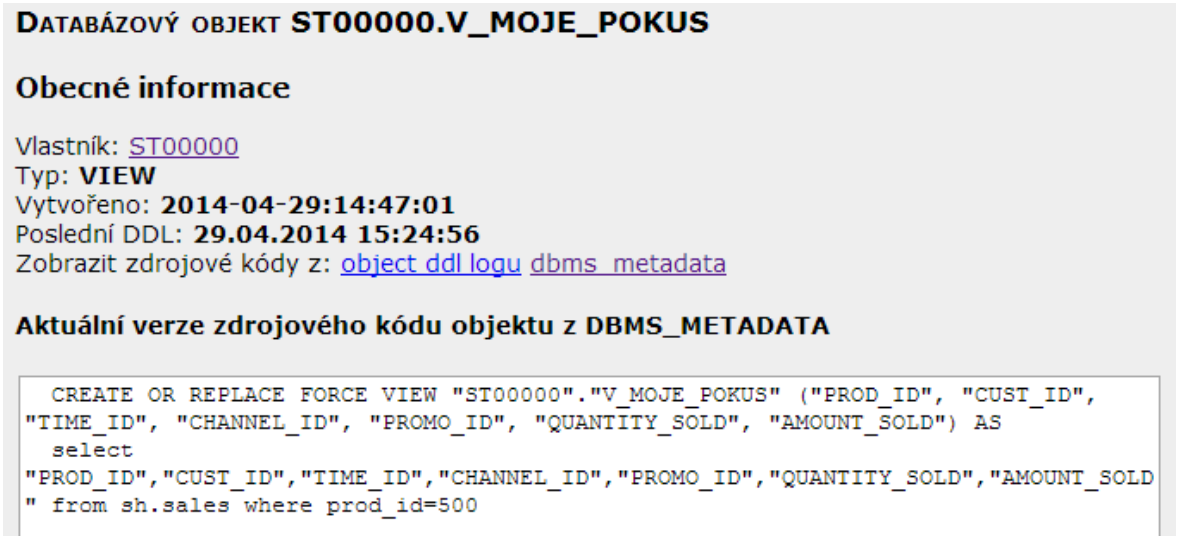
Obrázek 15 - Data objektu

3.4.10 Zdrojový kód objektu

Zdrojové kódy jsou k dispozici ze dvou možných zdrojů. Defaultní zdroj je ddl log (Obrázek 16), který je automaticky plněn pomocí triggeru. Druhým možným zdrojem je

využití balíčku DBMS_METADATA (Obrázek 17). První jmenovaný způsob umožňuje navíc i náhled do historie jednotlivých úprav databázového objektu. Toto platí pouze, pokud máme historická data k dispozici. Druhý jmenovaný způsob nám zobrazí pouze aktuální verzi DDL pro zvolený objekt.

Obrázek 16 - Zdrojový kód objektu včetně historie



Obrázek 17 - Zdrojový kód objektu generovaný pomocí DBMS_METADATA

Závěr

Cílem práce bylo vytvořit aplikaci, která by umožňovala zobrazení informací o činnosti uživatelů na databázovém serveru. Jako platforma byl zvolen databázový systém Oracle, a momentálně není možné přejít s aplikací i na jinou platformu, protože sběr dat z databázových serverů je velice individuální.

Zadaná práce byla splněna dle zadání a některé funkce byly dodány navíc. Během vývoje nedošlo k žádným velkým problémům. Celý vývoj vyžadoval studování dokumentace, pro sestavení dotazů, které tvoří hlavní základ celé aplikace. Většina informací je získávána právě pouze pomocí dotazů. Mezi tyto informace patří přihlašování uživatelů, seznamy databázových objektů, seznamy dotazů, obsahy cache a AWR reportů, dotazy z FGA, DDL příkazy atp. Další informace musely být sbírány pomocí triggerů, jako je historie databázových objektů atp. Grafické zpracování aplikace bylo koncipováno jako jednoduché a přehledné rozhraní, které je optimalizováno na funkčnost.

V teoretické části proběhla analýza všech zdrojů dat, které aplikace používá. Dále byla provedena rešerše v oblasti změn v databázovém systému Oracle 12c. Tato kapitola byla poměrně složitá, jelikož 99 % použitelných informací je k dispozici pouze v anglickém jazyce. Dalším problémem bylo vybrat, které změny v práci zmínit, protože celkový výčet novinek dosahuje k pěti stovkám.

V současné době je aplikace kompletní a plně funkční. To ale neznamená, že zde není ještě prostor pro další rozšíření. Již během vývoje se vyskytly logicky navazující celky, které by ještě aplikace mohla poskytovat. Mezi ně patří zdrojové kódy objektů, atp. Na další rozšíření ale už nezbyla kapacita a je tedy možné aplikaci rozšířit. Některé z těchto dalších požadavků vzniky během vývoje, po konzultaci s vyučujícími databázových systémů. Aplikace je totiž primárně určena právě pro podporu výuky databázových systému. K možných rozšířením tedy může patřit možnost úpravy FGA pravidel na libovolné objekty, nalezení objektů, které mají shodný zdrojový kód (plagiáty) atp. Prostor pro rozšíření je samozřejmě také v oblasti grafického zpracování aplikace. Velice podstatným rozšířením by také bylo provést průzkum v oblasti optimalizace získávání dat, protože v času odezvy je někdy poněkud pomalá.

Celkově si myslím, že aplikace má potenciál a po jejím dalším rozšíření může být opravdu velice užitečná. Během vývoje jsem se prakticky seznámila s vývojem pro databázový systém Oracle a celkově s architekturou, která mi připadá velice zajímavá a propracovaná. Práce mi rozšířila obzory i v oblasti anglické terminologie.

Použité zdroje

1. **SYS_CONTEXT.** *Oracle® Database SQL Language Reference.* [Online] Oracle. http://docs.oracle.com/cd/E11882_01/server.112/e41084/functions184.htm#SQLRF06117.

2. Kozák, Václav. Vývoj aplikací pod databáze Oracle. [Online] Únor 2014. http://www.plsql.cz/?category_name=novinky-oracle-12c.

3. SQL> /* Prompts */. *WITH_PLSQL hint - Oracle 12c New feature.* [Online] 2. Červen 2013. http://sqlprompts.blogspot.cz/2013/07/withplsql-oracle-12c-new-feature.html.

4. Identity Columns in Oracle Database 12c Release 1 (12.1). *ORACLE-BASE.* [Online] http://www.oracle-base.com/articles/12c/identity-columns-in-oracle-12cr1.php.

5. CREATE TABLE. *Oracle® Database SQL Language Reference.* [Online] 2013. http://docs.oracle.com/cd/E16655_01/server.121/e17209/statements_7002.htm#SQLRF01402.

6. Managing and Maintaining Time-Based Information. *Oracle® Database VLDB and Partitioning Guide.* [Online] 2014. http://docs.oracle.com/cd/E16655_01/server.121/e17613/part_lifecycle.htm#VLDBG14176.

7. Ensuring Application Continuity. *Oracle® Database Development Guide.* [Online] 2014. http://docs.oracle.com/cd/E16655_01/appdev.121/e17620/adfns_app_continuity.htm.

8. Deprecated and Desupported Features for Oracle Database 12c. *Oracle® Database Upgrade Guide.* [Online] Oracle, 2013. [Citace:] http://docs.oracle.com/cd/E16655_01/server.121/e17642/deprecated.htm#UPGRD60000.

9. DBA_AUDIT_TRAIL. *Oracle® Database Reference.* [Online] Oracle, 2009. http://docs.oracle.com/cd/B19306_01/server.102/b14237/statviews_3056.htm.

10. Memory Architecture. *Oracle® Database Concepts.* [Online] Oracle, 2005. http://docs.oracle.com/cd/B19306_01/server.102/b14220/memory.htm.

11. Automatic Workload Repository (AWR) in Oracle Database 10g. *ORACLE-BASE.* [Online] http://www.oracle-base.com/articles/10g/automatic-workload-repository-10g.php.

12. Security Enhancements In Oracle9i. *ORACLE-BASE.* [Online] http://www.oracle-base.com/articles/9i/security-enhancements-9i.php#FineGrainedAuditing.

13. DBMS_METADATA. *Oracle® Database PL/SQL Packages and Types Reference.* [Online] Oracle, 2007. http://docs.oracle.com/cd/B19306_01/appdev.102/b14258/d_metada.htm#BGBHHHBG.

14. ALL_SOURCE. *Oracle® Database Reference.* [Online] Oracle, 2009. http://docs.oracle.com/cd/B19306_01/server.102/b14237/statviews_2063.htm.

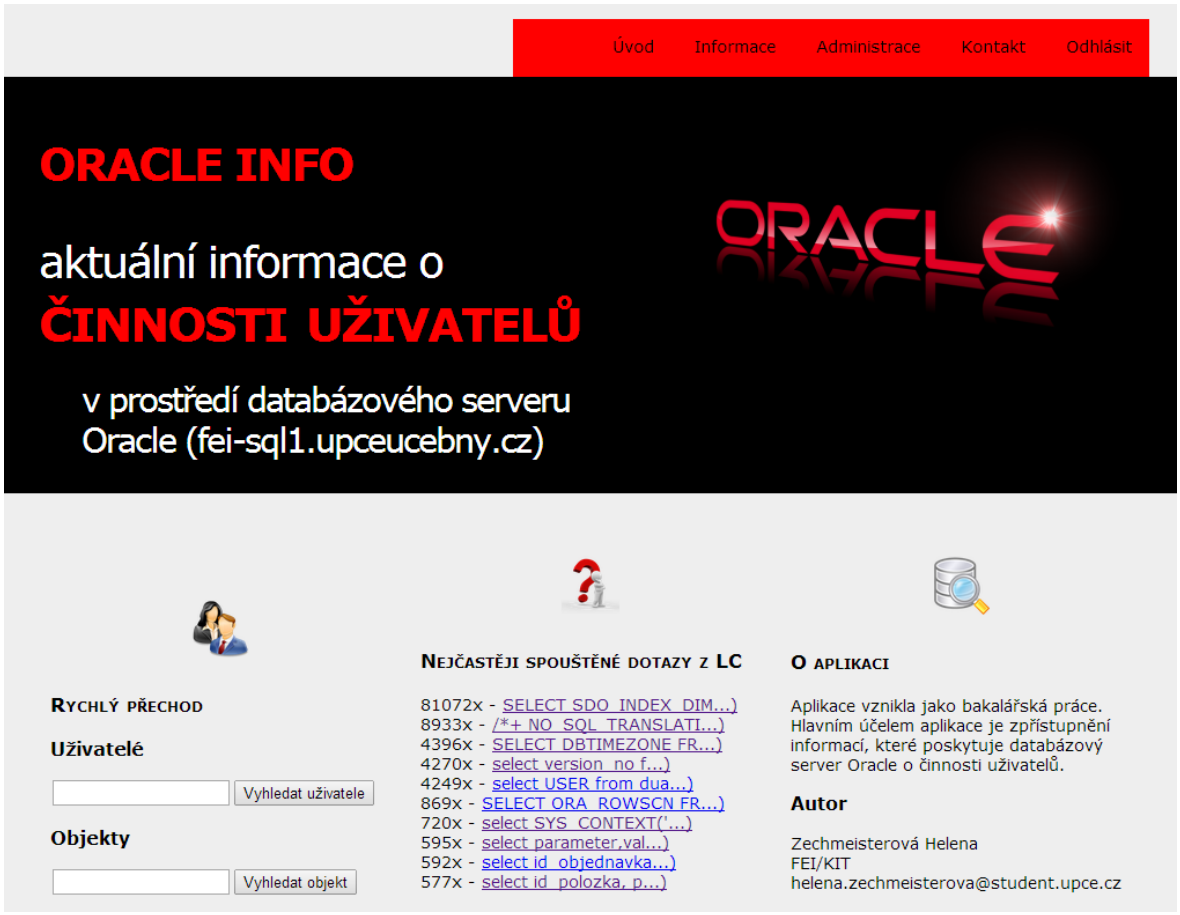
15. V$SQL. *Oracle® Database Reference.* [Online] 2009. http://docs.oracle.com/cd/B19306_01/server.102/b14237/dynviews_2113.htm.

16. V$SESSION. *Oracle® Database Reference.* [Online] Oracle, 2009. http://docs.oracle.com/cd/B19306_01/server.102/b14237/dynviews_2088.htm.

17. LUBBERS, Petr, Brian, ALBERTS a Frank, SALIM. *HTML5: programujeme moderní webové aplikace.* Brno : Computer Press, 2011. ISBN 987-80-251-3539-6.

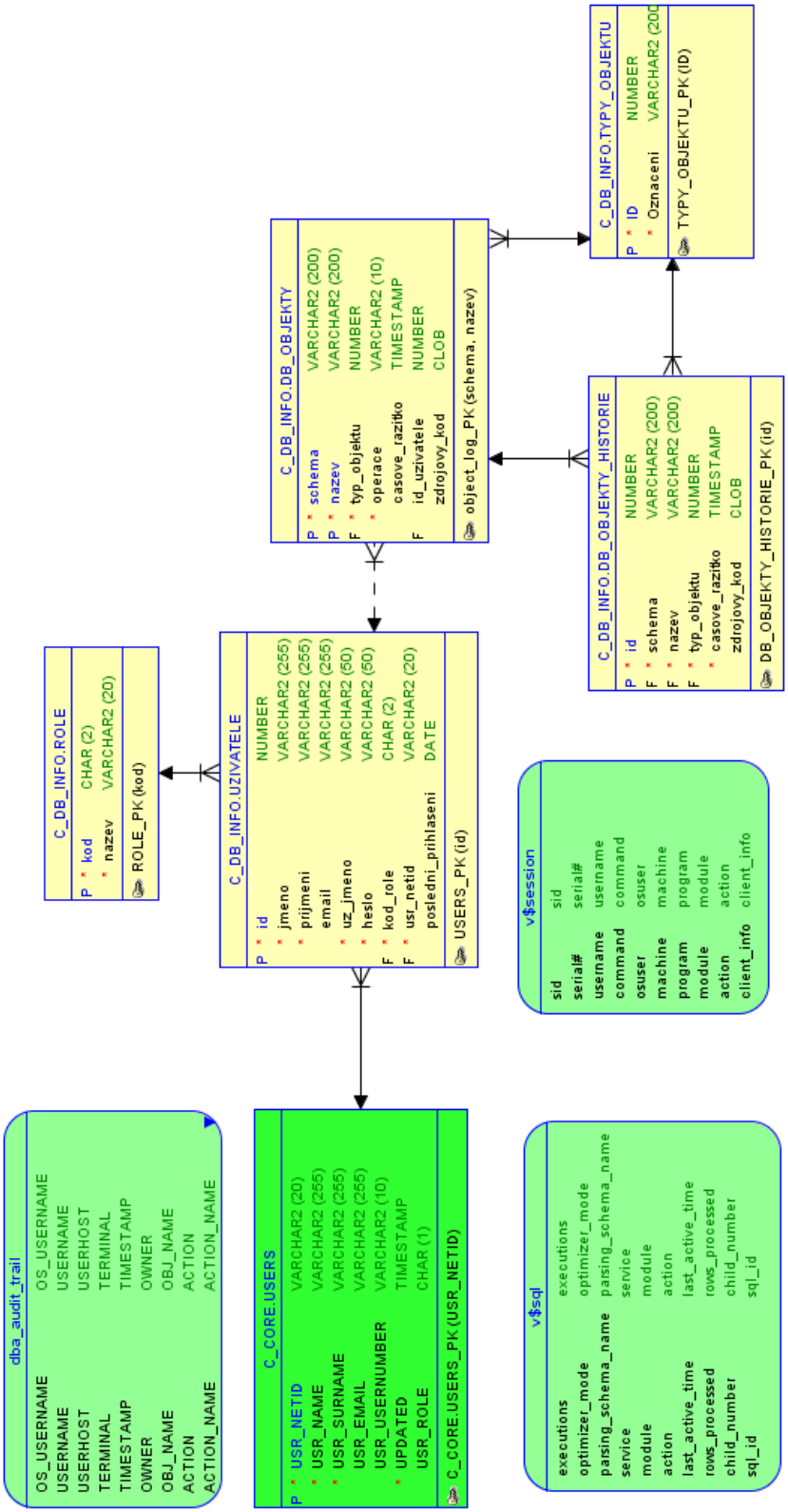
18. Lacko, L. *Oracle - Správa, programování a použití databázového systému.* místo neznámé : Computer Press, 2007. ISBN 978-80-25-490-2.

Příloha A – Úvodní obrazovka aplikace



Obrázek 18 - Úvodní obrazovka aplikace

Příloha B – Databázový model



Obrázek 19 - Databázový model