Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Nové technologie v Oracle Database 12c

Bakalářská práce

Autor: Jan Werner
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Barbora Tesařová, Ph.D.

Hradec Králové srpen 2016

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 16.8.2016 Jan Werner

Poděkování

Rád bych využil možnosti poděkovat na tomto místě Ing. Barboře Tesařové, Ph. D. za odborné vedení bakalářské práce, vřelý přístup, cenné rady a připomínky, kterými přispěla k vypracování této bakalářské práce.

Anotace

Předmětem této práce je představit nové technologie, které sebou přináší nová verze databázové platformy Oracle Database 12c. V teoretické části práce popisuje největší novinku tohoto databázového systému a to je multitenantní architektura. Dále bude představena architektura paměti a jako poslední věc bude podrobně vysvětlen pojem In-Memory. V praktické části pak budou představeny některé technologie probrané v teoretické části na praktických příkladech.

Klíčová slova:

Oracle, Database, 12c, multitenantní architektura, CDB, kontejnerová databáze, PDB, zásuvná databáze, SGA, PGA, In-Memory, In-Memory Column Store

Annotation

Title: New technologies in Oracle Database 12c

The object of this bachelor thesis is to describe new technologies that brings the new version of database platform called Oracle Database 12c. The theoretical part describes the greatest innovation of this database system and it is multitenant architecture. Additionally will be introduced memory architecture and at the end of the theoretical part will be explained the concept of In-Memory. The practical part will introduce some of the technologies discussed in the theoretical part on practical examples.

Keywords:

Oracle, Database, 12c, multitenant architecture, CDB, container database, PDB, pluggable database, SGA, PGA, In-Memory, In-Memory Column Store

Obsah

Úvod

Cílem této práce je seznámit se s novými technologiemi, které sebou přináší nová verze databázové platformy Oracle Database 12c. Oracle Database 12c, která byla vydána v létě roku 2013, přináší nové technologie z oblasti databázových systémů. Asi největším lákadlem této verze databázového systému je skutečnost, že databáze je připravena pro nasazení do cloudu. To ostatně také symbolizuje písmeno c v názvu této databázové platformy. Další novinkou této verze je zcela nová multitenantní architektura, která umožňuje, aby na jednom fyzickém serveru byla spuštěna pouze jedna databázová instance, kterou může sdílet mnoho databázových uživatelů. Poslední vydání Oracle Database 12c (12.1.0.2) navíc ještě doplňuje multitenantní architekturu novou komponentou paměti s názvem In-Memory Column Store. Tato technologie přináší nový způsob uchovávání dat v paměti, kdy se místo klasického řádkové formátu používá formát sloupcový. Toto vylepšení výrazně zvyšuje rychlost vykonávání SQL dotazů, speciálně pak analytických či statistických, kdy je potřeba procházet obrovské množství informací. Tato verze Oracle Database by tak měla ještě více než jindy pomáhat společnostem, které používají tento software, zjednodušit jejich práci při administraci, správě dat či maximalizaci zisku.

První část teoretické části se bude věnovat nové multitenantní architektuře. V souvislosti s multitenantní architekturou zde budou například vysvětleny termíny jako CDB, PDB a rozdíl mezi nimi. Druhá část teoretické části se bude věnovat správě paměti systému Oracle Database 12c. Na konci teoretické části bude představena technologie In-Memory.

V praktické části bude na ukázkových příkladech představena základní práce s multitenantní architekturou. Dále zde budou předvedeny základní úkony při správě paměti systému Oracle Database 12c. V poslední částí bude předvedeno jak pracovat s technologií In-Memory.

1 Teoretická část

Na počátku teoretické části práce bude stručně představena společnost Oracle. Bude zde uvedeno, kdo společnost založil a několik málo slov o její historii od samotného založení až po současnost. Dříve než budou popsány jednotlivé vybrané technologie, je důležité vysvětlit některé pojmy, které s danou problematikou úzce souvisí. Budou zde vysvětleny pojmy, jako je databázový systém nebo i například to, co se skrývá pod zkratkou SŘBD. Asi nejpřevratnější novinkou této verze Oracle Database je nepochybně nová multitenantní architektura, na které je tato nová verze databáze Oracle postavena. Multitenantní architektuře bude věnována první část teoretické části této práce. Zde bude například vysvětlen rozdíl mezi kontejnerovou a zásuvnou databází. Další část bude zaměřena na architekturu paměti. Budou zde popsány základní oblasti paměti i některé jejich důležité komponenty. Paměti se bude věnovat i poslední část této práce, kde bude podrobně popsána jedna z komponent SGA paměti. V souvislosti s tím zde bude vysvětlen nový termín „In-Memory“.

1.1 Oracle Corporation

Oracle Corporation musela ujít dlouhou cestu, než se z ní stala nadnárodní technická společnost, jak ji známe dnes. Společnost Oracle Corporation byla založena 16. června roku 1977 a za jejím vznikem stáli Larry Ellison, Bob Miner a Ed Oates. Tehdy ještě relativně malá společnost nesla název Software Development Laboratories. O dva roky později v roce 1979 se společnost přestěhovala do Kalifornie a byla přejmenována na Relational Systems Inc. V tomto roce byla uvolněna první relační databáze pro komerční účely. V roce 1981 se k této společnosti připojil Umang Gupta. Po sedmiletém působení ve společnosti IBM se nyní zasloužil o sepsání prvního podnikatelského plánu. Společnost byla znovu přejmenována, tentokrát na Oracle Systems Corporation. Následovalo přepsání, tehdy již databáze Oracle, do programovacího jazyka C, který měl širší škálu uplatnění. Zároveň byla uvolněna databáze Oracle v3. V roce 1984 vychází novější verze Oracle v4 ve které je již zavedena konzistence čtení dat. Databázový software byl přenesen na počítačovou platformu v listopadu téhož roku. O rok později vychází opět nová verze Oracle v5 s prvním relačním systémem řízení báze dat pro provoz v režimu klient-server. V roce 1987 byly představeny první Oracle aplikace založené na systému Unix.

O rok později vychází Oracle v6 s možností vytváření záloh a novou procedurální nadstavbou jazyka SQL, PL/SQL. Společnost se přesunula do svého nového sídla v Redwood Shores v Kalifornii a v roce 1989 již její příjem přesahoval 584 milionů dolarů. Oracle 7 vychází až v roce 1992, tentokrát již i s nástroji pro vývojáře, vylepšeným výkonem a mnoha dalšími novými prvky. U vzniku další verze Oracle databáze hrály hlavní roli internetové technologie. Další verze Oracle 8 byla představena v roce 1997. Tato verze zvládne pracovat s terabajty dat a využívá technologii SQL objektů. Zde byla také poprvé oznámena loajálnost s platformou Java. O rok později vychází také verze Oracle 8i databázového systému na platformě Linux. V roce 1999 společnost Oracle nabízí první systém řízení báze dat s podporou XML. Oracle 9i byla představena v roce 2000. V tomto roce Oracle zavadí Internet File System, který byl později přejmenován na Oracle Content Management. V roce 2003 společnost Oracle představuje Enterprise Grid Computing, který má být vydán následující rok. V roce 2004 společnost přebírá firmu PeopleSoft za přibližně 10 bilionů dolarů. Následující rok již společnost zaměstnává téměř 50 000 lidí. V dalších letech společnost nakupuje, ještě další firmy např. Agile Software Corporation, BEA Systems. V červenci roku 2005 vychází další verze Oracle 10g. Tato verze Oracle databáze se zaměřuje na Grid Computing, proto 10g. Předposlední vydanou verzí je Oracle 11g z roku 2007, která se velmi hojně využívá i dnes. V této verzi databázového systému byl největší důraz kladen na tzv. self-tuning, např. automatické rozložení zátěže na úložiště dat nebo automatická správa paměti. Nejnovější verze Oracle 12c byla vydána v létě roku 2013. Písmeno „c“ v názvu 12c vyjadřuje, že databáze je připravena pro nasazení do cloudu. [15]

1.2 Oracle Database 12c

Dříve než zde budou popsány ty nejpřevratnější novinky, které sebou přináší nová verze databázového systému Oracle Database 12c, bude zde vysvětleno několik termínů, které jsou důležité k pochopení následujících kapitol. Tato práce se bude věnovat novému databázovému systému Oracle Database 12c.

Databázový systém se skládá z databáze DB (ang. database) nebo také báze dat a tzv. systému řízení báze dat SŘBD (ang. database management system, DBMS). Databáze představuje centrálně zpracovávanou strukturu dat, pro kterou je vytvořena jediná interní organizace dat, společná pro všechny oblasti a způsoby využití těchto dat.

Centrální správa databáze, tzn. všechny implementační programy, jsou realizovány prostřednictvím speciálního programového vybavení, které se nazývá systém řízení báze dat. [16]

1.2.1 Multitenantní architektura

Verze databázového systému Oracle Database 12c disponuje především novou multitenantní architekturou. Klíčové slovo multitenantní (ang. multitenancy), které lze volně přeložit jako „více nájemná“ znamená, že na jednom fyzickém serveru běží pouze jedna aplikace, kterou sdílí více nájemníků. Těmito nájemníky mohou být konkrétní zákazníci, kteří chtějí přistupovat ke svým datům. Mnoho zákazníků sdílí stejné zdroje, proto je při návrhu aplikace velmi důležité jednotlivé zákazníky dobře zabezpečit a navzájem izolovat.

1.2.1.1 Kontejnerová a zásuvná databáze

Multitenantní architektura umožňuje Oracle databázi, aby fungovala jako multitenantní kontejnerová databáze (ang. Container database) označovaná jako CDB. Kontejnerová databáze nemusí obsahovat žádnou, jednu nebo mnoho zásuvných databází (ang. Pluggable database), označované jako PDB, vytvořené jednotlivými uživateli. Zásuvná databáze je kolekce schémat, objektových schémat a tzv. non-schema objektů (typy objektů, které nejsou obsaženy ve schématu např. role, uživatelé), která se zobrazí Oracle Net klientovi jako tzv. non-CDB. Všechny verze Oracle databází před Oracle Database 12c byly non-CDB. [1]

1.2.1.2 Logická struktura kontejnerové databáze

Pro lepší pochopení si zkusme představit, že kontejnerová databáze (CDB) je jakási mateřská loď, která obsahuje útočiště pro menší lodě (PDB). Mateřská loď jim poskytuje prostředky, o které se jednotlivé menší lodě musí dělit.

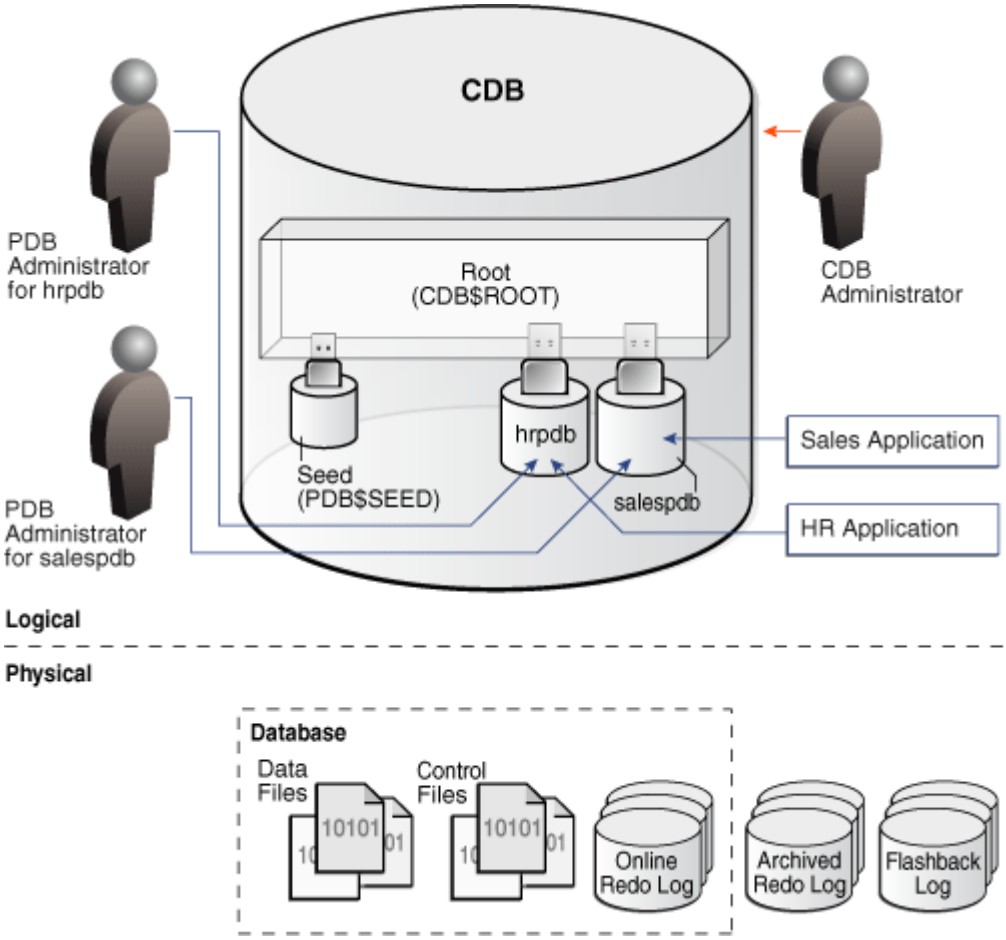
Kontejnerem v CDB je buď PDB nebo root. Root kontejner je kolekce schémat, objektových schémat a non-schémat, kterému patří všechny PDB. Každá kontejnerová databáze má následující kontejnery: [1]

- Root kontejner
  Root kontejner uchovává systémový datový slovník a tzv. „common“ uživatele. Systémový datový slovník bude popsán v kapitole 1.2.1.6. „Common“ uživatel je databázový uživatel známý v každém kontejneru. Root kontejner se také označuje jako CDB$ROOT.

- Seed PDB
  „Seed“ lze volně přeložit jako semínko. Jedná se o systémovou šablonu, kterou využívá kontejnerová databáze k vytvoření nových zásuvných databází. Tento kontejner je také označován jako PDB$SEED. Do tohoto kontejneru nelze přidávat objekty ani je upravovat.

- Žádný nebo více uživatelsky vytvořených zásuvných databází
  PDB je uživatelem vytvořená entita, která obsahuje data a kód potřebný pro specifickou množinu funkcí.

Následující obrázek č. 1 zobrazuje kontejnerovou databázi se čtyřmi kontejnery. Kontejnery CDB$ROOT a PDB$SEED jsou samozřejmostí. Jsou tu však i dvě zásuvné databáze, konkrétně „hrpdb“ a „salespdb“. Každá z těchto zásuvných databází má svou vlastní vyhrazenou aplikaci. Zásuvné databáze spravují každou zvlášť rozdílní PDB administrátoři. Napříč kontejnerovou databází existuje společný uživatel (ang. common user), který má jedinečnou identitu. Například společný uživatel SYS může spravovat CDB$ROOT a každou zásuvnou databázi.

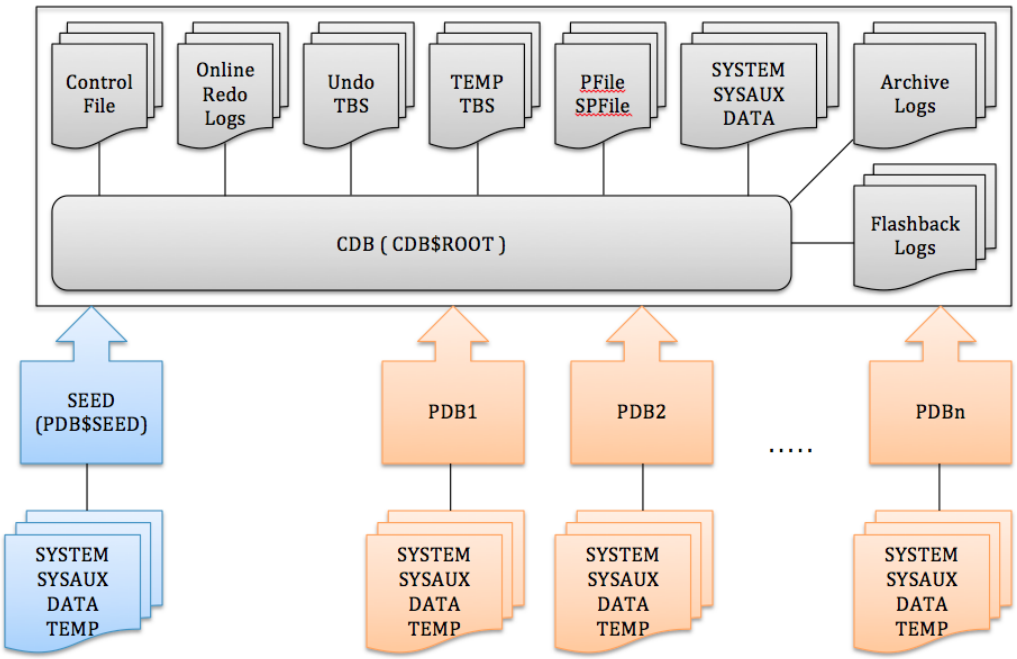


Obrázek 1: Ukázka CDB databáze, která obsahuje dvě PDB databáze (salespdb, hrpdb) (Zdroj: [1])

1.2.1.3 Fyzická struktura kontejnerové databáze

Jak již bylo zmíněno v předešlé kapitole, kontejnerová databáze může obsahovat mnoho zásuvných databází. Zásuvné databáze sdílejí zdroje poskytované kontejnerovou databází. Mezi tyto zdroje patří například paměť, procesy na pozadí, kontrolní soubor a UNDO a REDO soubory. Díky tomu může mnoho databází fungovat společně na jediné platformě, což ještě v předešlé architektuře Oracle 11gR2 nebylo možné. Všechny předešlé verze databází Oracle jsou označovány jako nekontejnerové. Tyto databáze se někdy také označují jako non-CDB. V Oracle 12c lze stále vytvářet nekontejnerové databáze a jejich administrace se v porovnání s databázemi 11g vůbec neliší.

Následující obrázek popisuje fyzickou strukturu kontejnerové databáze. Na obrázku č. 2 je vidět již známý root kontejner (CDB$ROOT), dále je zde vidět systémová šablona (PDB$SEED) pro vytváření nových zásuvných databází a konečně několik připojených zásuvných databází (PDB1 - PDBn). [2]



Obrázek 2: Fyzická struktura kontejnerové databáze (Zdroj: [2])

Jak již bylo řečeno, zásuvné databáze sdílejí některé zdroje poskytované kontejnerovou databází. Některé z těchto zdrojů zde budou rozebrány. Jedním z klíčových zdrojů, které poskytuje kontejnerová databáze je paměť. Architektuře a sdílení paměti se bude věnovat kapitola 1.2.2.

Dalším sdíleným zdrojem je kontrolní soubor (ang. control file). Kontrolní soubor je obsažen v každé databázi. Jedná se o malý binární soubor, který uchovává informace o fyzické struktuře databáze. V kontrolním souboru můžeme nalézt: [3]

- Název databáze
- Názvy a umístění přidružených datových souborů a tzv. redo log souborů
- Čas vytvoření databáze
- Současné LSN (Log Sequence number), každý záznam o provedení přihlášení k serveru dostane jedinečný identifikátor LSN
- Informace o checkpointech[1]

[1] Checkpoint je databázová událost, která synchronizuje modifikované bloky dat v paměti s datovými soubory na disku

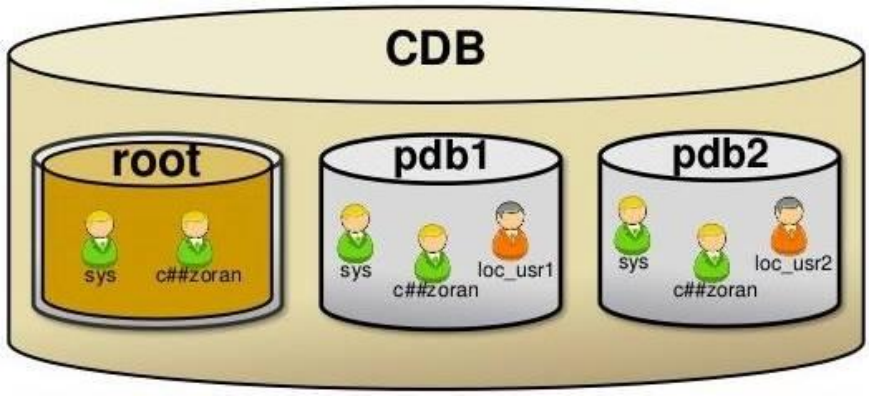
Kontrolní soubor musí být dostupný pro zapisování důležitých informací o databázi vždy, kdy je databáze otevřena. Bez kontrolního souboru nemůže být databáze nasazena a nahrazení je obtížné.
Nejzásadnější strukturou pro obnovovací operace je redo log. Ten se skládá ze dvou nebo více předem přidělených souborů, které uchovávají všechny změny provedené v databázových souborech a v kontrolním souboru. Jakmile je provedena změna v databázi, Oracle databáze vygeneruje redo záznam do redo zásobníku. Následně proces zvaný LGWR (Log Writer) vyprázdní obsah redo zásobníku do redo logu.[3]
Dalším zdrojem jsou Undo soubory. Tyto soubory využívají všechny zásuvné databáze. Jsou velmi důležité, protože uchovávají informace užitečné k navrácení provedených změn v databázi. Pokud bychom použili příkaz ROLLBACK, budou použity undo záznamy. Tyto záznamy vrátí zpět změny, které byly provedeny v databázi a mohou navrátit databázi do konzistentního stavu.
Dočasné soubory (ang. Temp files) obsahují objekty schémat pouze po dobu trvání relace. Jsou to speciální soubory užitečné k uchování dat v hashi, třídění a k jiným operacím. Naproti tomu permanentní tabulkové prostory seskupují trvalé objekty schémat. Segmenty pro tyto objekty jsou fyzicky uloženy v datových souborech. Tabulkové prostory SYSTEM a SYSAUX jsou defaultně obsaženy v kontejnerové databázi a každá zásuvná databáze má své vlastní.
SYSTEM je nezbytný administrativní tabulkový prostor, který je součástí databáze při jejím vytvoření. Databáze Oracle používají SYSTEM pro správu databáze. Tento tabulkový prostor obsahuje například datový slovník nebo tabulky a pohledy, které obsahují administrativní informace o databázi. Vše vlastní společný uživatel SYS.
SYSAUX jsou pomocné tabulkové prostory k SYSTEM tabulkovým prostorám. V těchto pomocných tabulkových prostorách jsou uchovávána metadata databáze, která nejsou umístěna v tabulkových prostorách SYSTEM. [4]

1.2.1.4 Uživatelé v kontejnerové databázi

V kontejnerové databázi existují dvě skupiny uživatelů. Společní uživatelé (ang. common users), o kterých již byla zmínka v kapitole 1.2.1.2, a místní uživatelé (ang. local users).
Společní uživatelé jsou uživatelé vytvoření v root kontejneru a jejich identita je stejná v root kontejneru a ve všech zásuvných databázích. Společní uživatelé se pak dále dělí na uživatele, které si může vytvořit sám uživatel, a na uživatele, které jsou

součástí databázové platformy Oracle Database. Například uživatel SYS, který je vidět na obrázku č. 3, je příkladem společného uživatele, který je součástí platformy Oracle Database. Při vytváření společného uživatele je třeba dodržovat určité zásady. Jednou z těchto zásad je, že jméno uživatele musí začínat znaky C## nebo c##. Uživatel c##zoran je příkladem takového uživatele.

Zatímco společní uživatelé působí ve všech kontejnerech, místní uživatele působí pouze v rámci jedné zásuvné databáze a nemají tak přístup k žádné jiné zásuvné databázi ani k root kontejneru. Příkladem nám může být uživatel „loc_usr1“, který působí pouze v rámci jedné zásuvné databáze, konkrétně v pdb1.



Obrázek 3: Uživatelé v kontejnerové databázi (Zdroj: [5]-upraveno autorem)

1.2.1.5 Uživatelské role v kontejnerové databázi

Stejně jako v případě uživatelů, existují v kontejnerové databázi společné a místní role. Společné role jsou užitečné pro operace napříč celým kontejnerem (ang. Cross-Container Operations). Tyto operace zajišťují, že každý společný uživatel bude mít tuto roli v každé zásuvné databázi, včetně root kontejneru. Pokud například nějaký kontejnerový administrátor vytvoří společného uživatele c##dba a přidělí mu společnou roli DBA, pak uživatel c##dba disponuje touto rolí v každé zásuvné databázi. Společné role se také dělí na role, které jsou již součástí Oracle Database a role vytvořené uživatelem. Příkladem již vytvořených rolí jsou role DBA a PUBLIC.

Naproti tomu místní role existují opět v rámci jedné zásuvné databáze. Jedná se o role a oprávnění, které lze uplatnit pouze v kontejneru, v kterém se nachází. Zásuvné databáze ve stejné kontejnerové databázi mohou obsahovat místní role stejného jména. Například místní role pdbadmin, vytvořená uživatelem, může existovat

v pdb1 i v pdb2. Tyto role jsou však na sobě zcela nezávislé, jako by existovali ve dvou oddělených nekontejnerových databázích.

1.2.1.6 Architektura datového slovníku

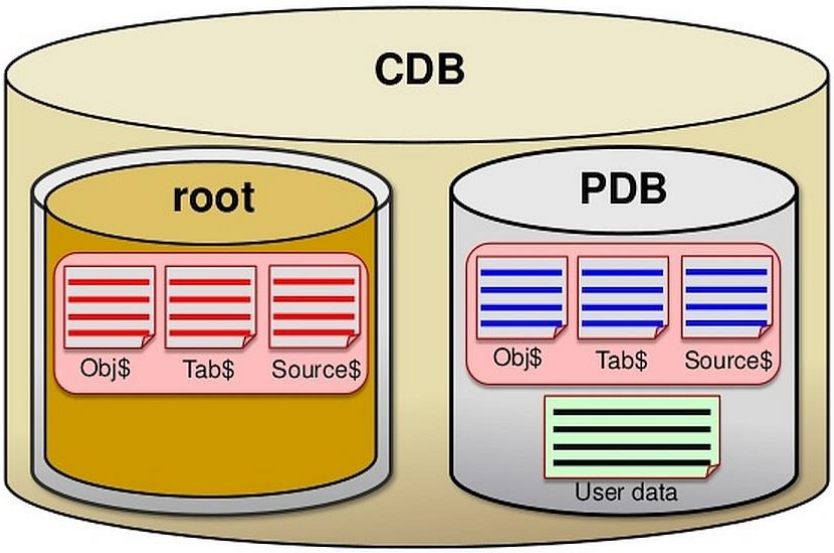
Důležitou součástí Oracle databáze je datový slovník. Jedná se o sadu tabulek a pohledů, které poskytují administrativní metadata o databázi. Pohledy pomocí joinů a klauzule WHERE dekódují data z tabulek do užitečných informací. Obsahují jména a popisky všech objektů v datovém slovníku. Některé pohledy jsou přístupné všem databázovým uživatelům a některé naopak jen administrátorům. V tabulkách můžeme nalézt následující informace: [6]

- Definice každého objektu schémat v databázi, včetně defaultních hodnot pro sloupce a informace o integritních omezeních
- Množství místa přiděleného a v současné době využívaného objekty schémat
- Názvy databázových uživatelů, oprávnění a rolí přidělených těmto uživatelům a další informace vztahující se k těmto uživatelům

1.2.1.6.1 Princip rozdělení datového slovníku

V kontejnerové databázi je datový slovník rozdělen na dvě části. První částí jsou systémová metadata, která jsou umístěná v root kontejneru a jsou sdílená pro všechny zásuvné databáze. Tyto metadata obsahují zejména objekty, které dodává Oracle, jako jsou definice tabulek datového slovníku a PL/SQL balíčky. Druhou částí databázového slovníku jsou uživatelská metadata. Každá zásuvná databáze má svá uživatelská metadata. [6]

Při připojení zásuvné databáze do root kontejneru dojde ke spojení těchto dvou částí a vznikne kompletní datový slovník. Nemusí tedy existovat mnoho duplicitních systémových metadat, stačí jedna, která budou sdílena všemi zásuvnými databázemi. Princip rozdělení datového slovníku lze ocenit i v případě, kdy bude potřeba změnit definici tabulky v datovém slovníku. Standardně by bylo třeba každou databázi aktualizovat zvlášť, s tímto principem však stačí aktualizovat systémová metadata v root kontejneru a problém je vyřešen.



Obrázek 4: Rozdělení datového slovníku (Zdroj: [5])

1.2.2 Architektura paměti databáze Oracle 12c

Paměť počítače představuje jednu z jeho klíčových komponent, bez kterých není možná jeho činnost. Čím více paměti váš počítačový systém má, tím lépe. Větší paměť pro počítač obvykle znamená i lepší výkon. Stejně jako váš domácí počítač i databázový systém, k tomu aby vůbec mohl být spuštěn, potřebuje dostatečně velké množství paměti.

Při spuštění instance databázového systému se alokuje oblast paměti a spustí se tzv. background procesy (procesy na pozadí). Oblast paměti si drží informace, jako jsou následující: [7]

- Programový kód
- Informace o každé připojené relaci
- Informace potřebné během spouštění programu (např. současný stav dotazu)
- Informace, které jsou sdíleny mezi jednotlivými procesy (např. údaje o zámcích)
- Cache data (např. redo záznamy, které ještě nejsou uloženy na disku)

Architektura paměti Oracle Database se skládá z několika základních paměťových oblastí, kde každá oblast se dále skládá z mnoha komponent či podoblastí. Zde budou popsány jen ty základní oblasti paměti. Podle společnosti Oracle se mezi základní paměťové struktury řadí: [7]

- SGA (ang. System global area)
- PGA (ang. Program global area)
- UGA (ang. User global area)
  Oblast paměti spojená s uživatelskou relací, která obsahuje informace o klientovi (přihlašovací informace, IP adresu, atd.). V případě kdy se uživatelé připojují k dedikovanému serveru (1 uživatel = 1 serverový proces) je UGA součástí paměti PGA. V opačném případě kdy se uživatelé připojují ke sdílenému serveru (každý proces může obsluhovat více uživatelů) je UGA paměť součástí SGA paměti.
- Oblasti softwarových kódů

1.2.2.1 SGA - Globální systémová oblast

Globální systémová oblast je skupina sdílených paměťových struktur, označovaných také jako SGA komponenty, které obsahují data a kontrolní informace o jedné databázové instanci. SGA je paměť pro čtení i zápis a je společná pro všechny servery a procesy na pozadí. Někdy je tato oblast označovaná jako globální sdílená oblast (ang. Shared global area). Nejdůležitějšími komponenty SGA jsou například: [7]

- **Database Buffer Cache** – Udržuje kopie datových bloků načtených z datových souborů. Při žádosti o data se Oracle nejprve podívá do database buffer cache a pokud je zde najde (cache hit), jsou data použita. V opačném případě musí být načtena z disku. Všechny uživatelské procesy připojené k databázové instanci sdílejí přístup k database buffer cache.
- **In-Memory Column Store** – Tento komponent a zároveň další stěžejní vlastnost nového databázového systému bude popsán v kapitole 1.2.3.3.

- **Redo Log Buffer** – O redo logu již byla řeč v kapitole 1.2.1.3.
- **Shared Pool** – Tato sdílená oblast ukrývá různé typy programových dat, jako jsou například PL/SQL procedury a balíčky, systémové parametry a jiné. Shared pool je zapojen téměř při každé operaci, která se v databázi vyskytuje. Například pokud uživatel databáze provede SQL příkaz, pak databáze Oracle přistupuje k této sdílené oblasti.
- **Large Pool** – Large pool je volná oblast paměti pro rozsáhlé paměťové alokace, jako jsou obnova či záloha.
- **Java Pool** – Jedná se o oblast paměti, která uchovává všechny relace spojené s programovacím jazykem Java a data v rámci Java Virtual Machine (JVM).
- **Streams Pool** – Uchovává fronty zpráv v zásobnících a poskytuje paměť pro Oracle Streams. Tento proud zpráv zajišťuje sdílení informací v databázi nebo z jedné databáze do jiné.
- **Fixed SGA** – Obsahuje základní informace o stavu databáze, informace proudící mezi procesy, jako jsou informace o zámcích. Velikost této oblasti paměti je defaultně nastavena a nemůže být manuálně změněna.

Velikost SGA paměti je určena inicializačními parametry. Můžeme ji měnit i za běhu databázové instance. Celková velikost všech komponent SGA paměti však nesmí překročit hodnotu nastavenou v parametru SGA_MAX_SIZE.
Paměť pro všechny dynamické komponenty SGA paměti se uvádí v jednotkách zvaných jako „zrníčka“ (ang. granules). Velikost jednoho zrníčka se odráží na velikosti SGA paměti požadované při startu databázové instance. [9]

Tabulka 1: Velikost zrníčka v závislosti na velikosti SGA paměti (Zdroj: [8])

Velikost SGA paměti	Velikost zrníčka
Menší nebo rovno 1GB	4MB
Větší než 1GB a menší nebo rovno 8GB	16MB
Větší než 8GB a menší nebo rovno 16GB	32MB
Větší než 16GB a menší nebo rovno 32GB	64MB
Větší než 32GB a menší nebo rovno 64GB	128MB
Větší než 64GB a menší nebo rovno 128GB	256MB
Větší než 128GB	512MB

Rozdělení paměťového prostoru pro jednotlivé komponenty SGA paměti znázorňují následující tabulky č. 2 a č. 3. Parametr SGA_MAX_SIZE je v tomto případě nastaven na 128 megabajtů. Velikost zrníčka je tedy podle tabulky č. 1 nad tímto textem rovna čtyřem megabajtům. Moduly Fixed SGA a Redo Buffer nejsou brány v potaz, jelikož se jedná o fixní komponenty a jejich velikost se tedy nemění. Nicméně jsou to také komponenty SGA paměti, tak že jsou v tabulkách zaznamenány také.

Tabulka 2: SGA paměť, rozložená na jednotlivá zrníčka (Zdroj: [9])

Tabulka 3: Ukázka velikosti jednotlivých komponent SGA paměti (Zdroj: [9])

4MB	Fixed SGA
16MB	Volná paměť
48MB	Database Buffer Pool
20MB	Java Pool
12MB	Large Pool
32MB	Shared Pool
4MB	Redo Buffer

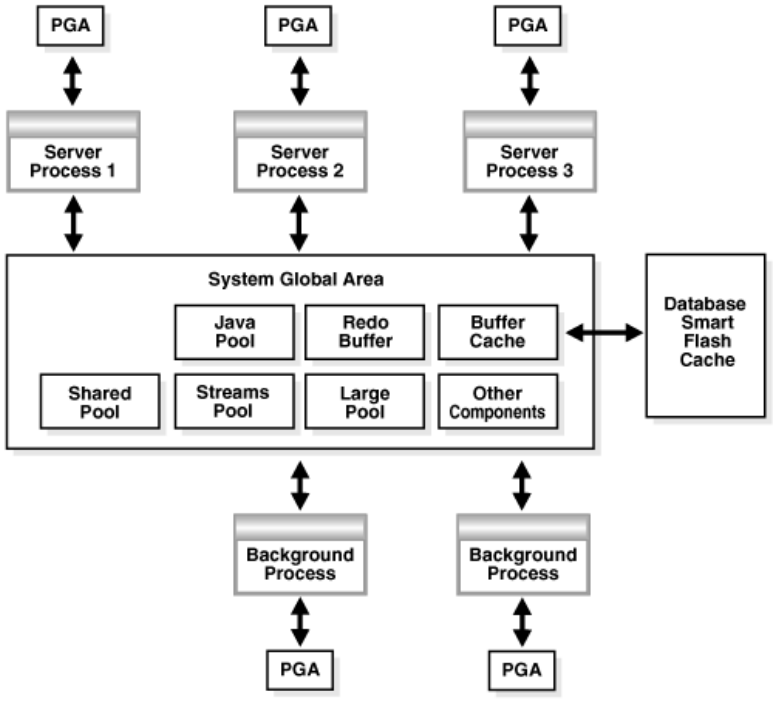
1.2.2.2 PGA - Globální programová oblast

PGA je oblast paměti, která obsahuje data a kontrolní informace pro serverové procesy. Jedná se o nesdílenou paměť vytvořenou databází Oracle při startu serverového procesu. Přístup k této paměti je exkluzivní jen pro serverové procesy. Pro každý serverový proces existuje vždy jedna PGA paměť. Pokud se tedy k jedné

databázové instanci připojí 100 uživatelů, musí být alokováno 100 PGA pamětí pro jednotlivé procesy. I procesy na pozadí alokují své vlastní PGA paměti.

„Celková instanční PGA paměť" je celková přidělená paměť pro všechny procesy na pozadí a serverové procesy, které jsou připojené k instanci databáze Oracle. Kolekce všech individuálních PGA se nazývá „PGA instance". (přeloženo autorem).[7]

Následující obrázek ještě více dokresluje rozdíl mezi probranými oblastmi paměti. Jak si můžete všimnout, na obrázku chybí jedna komponenta SGA paměti. Jedná se o komponentu „In Memory Column Store". Je to způsobené tím, že funkce In-Memory byla představena až ve verzi 12.1.0.2, kdežto tento obrázek pochází ze základní verze databáze Oracle 12c. Proto se zde tato komponenta ještě nevyskytuje.



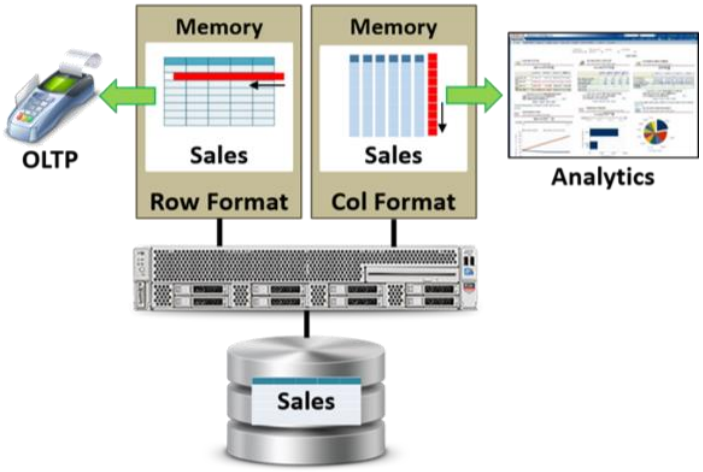
Obrázek 5: Architektura paměti databáze Oracle 12c (Zdroj: [8])

1.2.3 In-Memory

Relační databáze uchovávají data v klasickém řádkovém formátu. To znamená, že každá nově provedená transakce nebo záznam ukládaný do databáze je reprezentován novým řádkem v tabulce. Řádek neboli záznam je tvořen několika sloupci, kde v každém sloupci je uvedena odlišná informace o daném záznamu. Řádkový formát je vhodný pro systémy provádějící online transakce (OLTP), neboť umožňuje rychlý přístup ke všem sloupcům daného záznamu. Tento princip

ukládání dat není nikterak špatný, avšak může nastat situace, kdy je vhodnější upřednostnit jiný způsob uchovávání dat.

Nové vydání databáze Oracle 12c (12.1.0.2) přináší převratnou technologii uchovávání dat ve sloupcovém formátu. Princip spočívá v tom, že jednotlivé atributy transakce nebo záznamu jsou uloženy do separátních sloupcových struktur. Tyto sloupcové struktury jsou uloženy v komponentě SGA paměti zvaném „In-Memory Column Store“. Tento komponent nijak nenahrazuje buffer cache, ale vystupuje jako doplněk. Což znamená, že oba komponenty uchovávají stejná data, ale v rozdílných formátech.[10]



Obrázek 6: Ukázka formátů zpracování dat v databázi Oracle 12c (řádkový formát, sloupcový formát) (Zdroj: [11])

Tento princip ukládání dat je vhodný zejména pro analytické či statistické úlohy, kde je často obrovské množství záznamů a jen několik málo atributů. Provedení dotazu může v případě řádkového zpracování dat zabrat i několik hodin. Pokud však je k dispozici sloupcové zpracování dat, může provedení identického dotazu zabrat jen pár vteřin.

1.2.3.1 Analytické dotazy

Jedná se o skupinu SQL dotazů, které vyhledávají v rozsáhlých databázových tabulkách jen určité podmnožiny dat. Výsledná data z těchto dotazů mají velký význam. Pro příklad si představme, že existuje podnikatel, který vlastní určitý obchodní řetězec.

Tohoto podnikatele by například mohlo zajímat, které produkty mu poskytují nejvyšší zisk, nebo by chtěl mít seznam deseti produktů, které se nejvíce prodávají. Takové informace by pro něho měli zásadní význam v tom, aby věděl, kterým směrem se má dále vydat. Analytické dotazy se vyznačují tím, že:

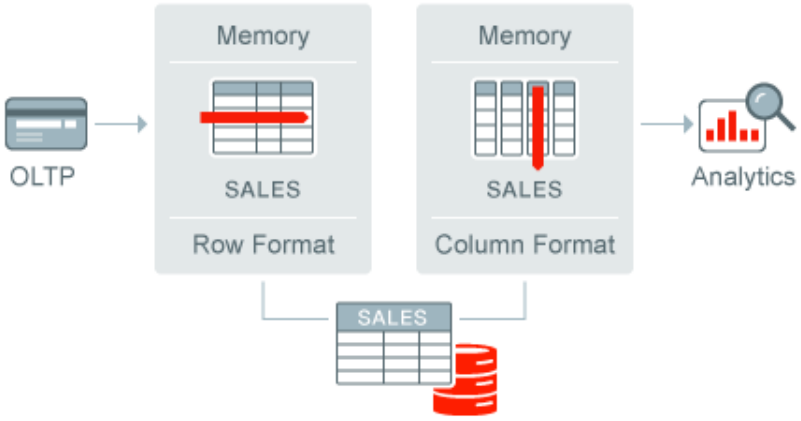
- Skenují velké množství dat, vybírají určité podmnožiny dat z rozsáhlých databázových tabulek
- Používají predikáty pro filtrování dat (=, <, >, BETWEEN atd.)
- Používají spojování tabulek, které redukuje výsledné množství dat a kombinuje data z více tabulek
- Obsahují operátory pro komplexní výpočty (AVG, MAX, MIN, SUM) nebo řazení (GROUP BY)

1.2.3.2 Řádkový vs Sloupcový formát databáze

Před více než třiceti lety se společnost Oracle rozhodla, že bude ukládat data do databázových tabulek v řádkovém formátu. S řádkovým formátem, každý nový záznam přidaný do tabulky, představuje nový řádek v tabulce. Tento řádek je složen z mnoha sloupců, kde každý z těchto sloupců představuje jiný atribut (informaci o záznamu). Například přidáním nového studenta do databáze, bude nový záznam resp. nový řádek složen ze sloupců, kde si první sloupec bude držet jméno studenta. V dalším sloupci může být uchovávána například adresa nebo email a tak podobně. Řádkový formát uchovávání dat je velmi efektivní pro OLTP transakce. Což jsou operace typu přidávání nového studenta, vytváření nové objednávky, aktualizace údajů o určitém zákazníkovi nebo smazání zákazníka. Alternativou k řádkovému formátu je, díky nové technologii In-Memory, formát sloupcový. V databázi, kde se používá sloupcový formát pro uchovávání dat, jsou jednotlivé atributy daného záznamu uloženy do separátních sloupcových struktur. Sloupcový formát je velmi efektivní při práci s analytickými dotazy, kde se často prohledává celá databáze a kde je ve výsledku potřeba jen určitá podmnožina dat. V době před třiceti lety se Oracle určitě rozhodl správně, když zvolil řádkový formát pro uchovávání dat v databázových tabulkách. Avšak v dnešní době, právě pro účely kladení

analytických či statistických dotazů na robustní databázové systémy se hodí spíše sloupcový formát uchovávání dat.

Zřejmě proto se Oracle rozhodl, že přistoupí ke kompromisu a vytvoří databázi, kde bude možné uchovávat data současně jak v řádkovém tak i sloupcovém formátu. Vznikla tzv. databáze dvojího formátu (ang. Dual Format Database).



Obrázek 7: Ukázka tzv. Dual Format Database (Zdroj: [13])

Pokud tedy nyní přijde do databáze Oracle SQL dotaz, nejprve se rozhodne, zda se jedná o čistě analytický dotaz, potom bude tento dotaz odeslán ke zpracování pomocí sloupcového formátu. Jde-li spíše o čistě transakční dotaz, v takovém případě bude dotaz směřovat do komponenty Buffer Cache a bude klasicky zpracován pomocí řádkového formátu. Co je však na tomto přístupu jedinečné, je to, že na počítačovém disku je uložen pouze jeden formát a není tak vyžadován žádný další diskový prostor pro uložení dat v druhém formátu. Data uložená na disku počítače jsou zpracovány pomocí tradičního řádkového formátu a stejně tak nástroje pro správu databáze Oracle vyžadují, aby byla data uložena v řádkovém formátu. Data jsou navíc transakčně konzistentní, což znamená, že jakmile se data změní, tak se tyto změny projeví jak v sloupcovém, tak i v řádkovém formátu. [10]

1.2.3.3 In-Memory Column Store

In-Memory Column Store nebo také zkráceně IMCS je volitelná, statická oblast SGA paměti, která ukládá kopie databázových tabulek ve speciálním sloupcovém

formátu, který je optimalizován pro rychlé skenování dat. Databázové objekty, které mají být zpracovány pomocí IMCS, musí být označeny klíčovým slovem INMEMORY. Tímto klíčovým slovem lze označit následující databázové objekty:[12]

- Sloupec tabulky
- Tabulka
- Materializovaný pohled
- Tabulkový prostor
- Oddíl

Pokud je označení klíčovým slovem INMEMORY provedeno na úrovni tabulkových prostor, pak všechny tabulky a materializované pohledy spadající do těchto tabulkových prostor jsou označeny také. Sloupcový formát existuje jen v paměti databáze. IMCS je defaultně vypnutý. Pro povolení IMCS je nutné nastavit inicializační parametr s názvem INMEMORY_SIZE na nenulovou hodnotu.[12]
Kde a jak nastavit tento parametr nebo, které tabulky označit pro zpracování sloupcovým formátem bude ukázáno v praktické části této práce.

1.2.3.4 Kompresní mechanismy

Tradiční komprese databázových tabulek je považována za mechanismus, který šetří místo na disku. Databázové tabulky, které jsou ukládány do komponenty In-Memory Column Store, jsou komprimovány pomocí nové sady kompresních algoritmů, které nejen, že pomáhají šetřit místo na disku, ale zaměřují se také na zvýšení rychlosti vykonávání dotazů. Nový kompresní formát, který přichází s technologií Oracle In-Memory umožňuje dotazům komunikovat přímo s požadovanými komprimovanými sloupci. To znamená, že operace typu skenování či filtrování budou prováděny nad mnohem menším množstvím dat. Data jsou dekomprimována jen tehdy, mají-li být součástí sady výsledků – odpovědi na daný dotaz. Kompresi lze nastavit pomocí klíčového slova MEMCOMPRESS, sub-klauzule atributu INMEMORY. Existuje celkem šest úrovní komprese, kde každá poskytuje různý stupeň komprese a výkonu.[10]

Všechny možností nastavení komprese ukazuje tabulka č. 4.

Tabulka 4: Ukázka všech kompresních úrovní, které lze nastavit databázovému objektu (Zdroj: [10])

NO MEMCOMPRESS	Data jsou nahrávána bez komprese
MEMCOMPRESS FOR DML	Minimální komprese, optimalizováno pro DML operace
MEMCOMPRESS FOR QUERY LOW	Optimalizováno na výkon při dotazování (defaultní nastavení)
MEMCOMPRESS FOR QUERY HIGH	Optimalizováno jak na výkon při dotazování tak i pro úsporu místa
MEMCOMPRESS FOR CAPACITY LOW	Vyvážená komprese se zaměřením spíše na úsporu místa
MEMCOMPRESS FOR CAPACITY HIGH	Optimalizováno pro úsporu místa

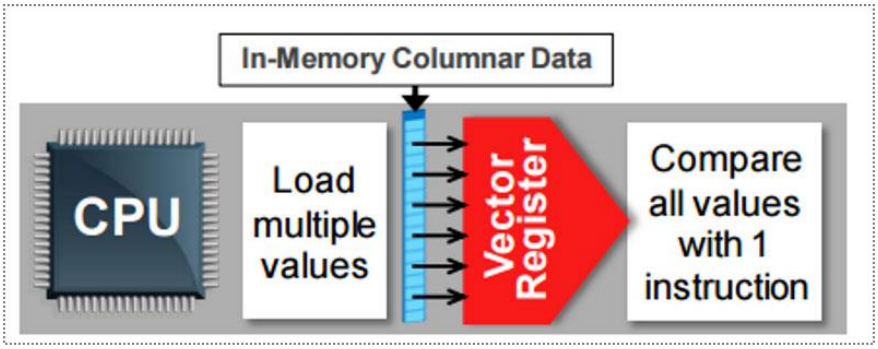
Defaultně jsou data komprimována pomocí kompresní úrovně FOR QUERY LOW, která poskytuje nejlepší výkon pro zpracování dotazů. Kompresní poměr se může lišit dvakrát až dvacetkrát, v závislosti na zvolené kompresní úrovni, typu dat a obsahu tabulky. Kompresní techniku lze měnit v rámci sloupce či oddílu dané tabulky. Některé sloupce je možné optimalizovat pro rychlost skenování a jiné pro úsporu místa na disku. [10]

1.2.3.5 In-Memory Storage Index

Další redukce množství procházených dat je možná díky In-Memory indexům skladování, které jsou vytvářeny a spravovány automaticky na každém sloupci v komponentu In-Memory Column Store. Indexy skladování umožňují omezovat data na základě filtračních predikátů, které jsou součástí příchozího SQL dotazu. Indexy skladování uchovávají minimální a maximální hodnotu pro každý sloupec IMCU (In-Memory Compression Unit). Jestliže příchozí dotaz obsahuje klauzuli WHERE, pomocí indexů skladování započne prohledávání těchto IMCU a porovnávání minimální a maximální hodnoty s hodnotou specifikovanou v SQL dotazu. Pokud je hodnota specifikovaná SQL dotazem mimo rozsah minima a maxima daného IMCU, tak ke skenování tohoto IMCU vůbec nedojde.[10]

1.2.3.6 Vektorové zpracování SIMD

Pro data, která jsou vybrána pro skenování, používá databáze vektorové zpracování SIMD (Single Instruction processing Multiple Data values). Namísto posuzování každé položky v daném sloupci, jedné za druhou, vektorové zpracování SIMD umožňuje, aby celá sada hodnot ve sloupci byla posouzena společně v rámci jedné instrukce procesoru.[10]



Obrázek 8: SIMD vektorové zpracování (Zdroj: [14])

Sloupcový formát užívaný komponentou In-Memory Column Store je speciálně navržen tak, aby maximalizoval počet položek v daném sloupci, které mají být nahrány do vektorových registrů procesoru a zpracovány jedinou procesorovou instrukcí. Díky vektorovému zpracování SIMD lze v databázi Oracle 12c skenovat miliardy řádků za sekundu. Pokud bychom za stejných podmínek použili klasický řádkový formát, byla by rychlost zpracování téměř stokrát pomalejší.

1.2.3.7 In-Memory Column Store – Plnění dat

Jednou z dalších možností jak zrychlit vykonání požadovaného SQL dotazu, je nastavit tzv. úroveň priority. Pokud je již rozhodnuto, že danému databázovému objektu bude umožněno, aby byl naplněn do komponenty IMCS, jsou dvě možnosti jak postupovat dál. První z možností je zároveň možností defaultní. Jedná se o to, že si databáze Oracle bude sama řídit naplňování databázových objektů do IMCS. Druhou možností je manuálně specifikovat úroveň priority, která bude určovat prioritu databázového objektu ve frontě naplňování do IMCS.

SQL klauzule s názvem INMEMORY PRIORITY umožňuje větší kontrolu nad pořadím naplňování. Mohou například existovat databázové objekty, které se často používají pro analytické dotazy. Těmto objektům musí být nastavena vyšší priorita při plnění dat do IMCS. Úrovně priorit, které lze nastavit databázovým objektům, ukazuje tabulka č. 5.[10]

Tabulka 5: Ukázka všech úrovní priority, které lze nastavit databázovému objektu (Zdroj: [10])

PRIORITY CRITICAL	Objekt je naplněn okamžitě, jakmile je databáze otevřena.
PRIORITY HIGH	Objekt je naplněn až po té, co jsou naplněny všechny objekty s prioritou CRITICAL, pokud je v IMCS ještě dost místa.
PRIORITY MEDIUM	Objekt je naplněn až po té, co jsou naplněny všechny objekty s prioritou CRITICAL a HIGH, pokud je v IMCS ještě dost místa.
PRIORITY LOW	Objekt je naplněn až po té, co jsou naplněny všechny objekty s prioritou CRITICAL, HIGH a MEDIUM, pokud je v IMCS ještě dost místa.
PRIORITY NONE	Jen objekty, které jsou první naskenovány (výchozí), pokud je v IMCS dost místa.

Pokud má více než jeden databázový objekt nastavenu jinou úroveň priority než NONE, pak všechny data z těchto objektů budou naplňovány do IMCS na základě úrovní priorit. Nejdříve jsou naplněna data databázových objektů s úrovní priority CRITICAL, dále jsou naplněna data s úrovní priority HIGH a tak dále. Pokud již není dostatek volného místa pro naplnění dalších objektů, žádné další objekty nejsou do IMCS naplněny dokud se nějaký nový prostor neuvolní. Po restartování databázové instance jsou všechna data s úrovní priority různé od NONE naplněna do IMCS. Úroveň priority musí být nastavena celé tabulce či celému oddílu tabulky. Specifikování různých úrovní priority pro různé sloupce v tabulce není povoleno.[10]

2 Praktická část

V praktické části práce budou některé vybrané funkcionality, které byly popisovány v části teoretické, předvedeny na praktických příkladech. Nejprve zde budou představeny užitečné nástroje pro správu databázových systémů Oracle. Dále zde bude ukázáno jak postupovat při instalaci databáze Oracle 12c. V rámci instalace databáze bude vytvořen kontejner CDB, který bude později obsahovat několik zásuvných databází. V další části bude předvedeno, jak se s těmito zásuvnými databázemi pracuje. Předposlední část se bude věnovat správě paměti a In-Memory. Například zde bude uveden rozdíl v použití komponenty Buffer Cache a In-Memory Column Store. Na konci praktické části bude uvedeno několik užitečných příkladů, hodících se při správě databází Oracle. Praktické ukázky budou prováděny v prostředí Oracle SQL Developer či nástroje SQL*Plus a to za pomocí dotazovacího jazyka SQL či procedurálního jazyka PL/SQL.

2.1 Nástroje pro správu databáze Oracle

Součástí instalačního balíčku databáze Oracle 12c jsou i některé základní nástroje, pomocí kterých lze s databází efektivně pracovat. Těmito nástroji jsou zejména:

- Oracle SQL Developer
  Jedná se o integrované vývojové prostředí, které je, zejména díky grafickému rozhraní uživatelsky přívětivější, zjednodušuje vývoj a správu databáze Oracle, ať už v tradičním či cloudovém nasazení. Navíc je tento nástroj zdarma a volně dostupný na webových stránkách společnosti Oracle.
- SQL*Plus
  SQL*Plus je interaktivní a dávkovací nástroj pro dotazování, který je součástí každého databázového systému Oracle. S využitím dotazovacího jazyka SQL či procedurálního jazyka PL/SQL lze pomocí tohoto nástroje spravovat a vyvíjet databázi Oracle. Tento nástroj je velmi podobný příkazové řádce z prostředí Windows.

- Oracle Database Configuration Assistant (DBCA)
  Tento nástroj jak již název napovídá, slouží k vytváření a konfiguraci databází Oracle. Seznámit se s ním je možné již při instalaci databáze Oracle, kde s jeho pomocí lze vytvořit a následně nakonfigurovat novou databázovou instanci.
- Oracle Administration Assistant for Windows
  Tento pomocný nástroj umožňuje sledovat procesy a služby databáze Oracle v prostředí Windows. Zde je také možné restartovat databázovou instanci pomocí grafického rozhraní.

2.2 Instalace Oracle Database 12c, vytvoření a konfigurace CDB

Ještě před tím, než bude možné se pustit do instalace databáze Oracle 12c na 64 bitový operační systém Windows 7, je třeba upozornit na minimální hardwarové požadavky, které musí daný stroj splňovat, aby bylo možné efektivně pracovat s databází Oracle 12c. Minimální požadavky na počítačový hardware shrnuje tabulka č. 6.

Tabulka 6: Minimální hardwarové požadavky pro instalaci produktu Oracle Database 12c (Vlastní tvorba)

Hardwarová komponenta	Minimální hodnota
Systémová architektura	Procesory: AMD64 a Intel EM64T
Fyzická paměť (RAM)	Min. 4 GB
Místo na disku	10 GB
Video adaptér	256 barev
Rozlišení obrazovky	Min. 1024 x 768

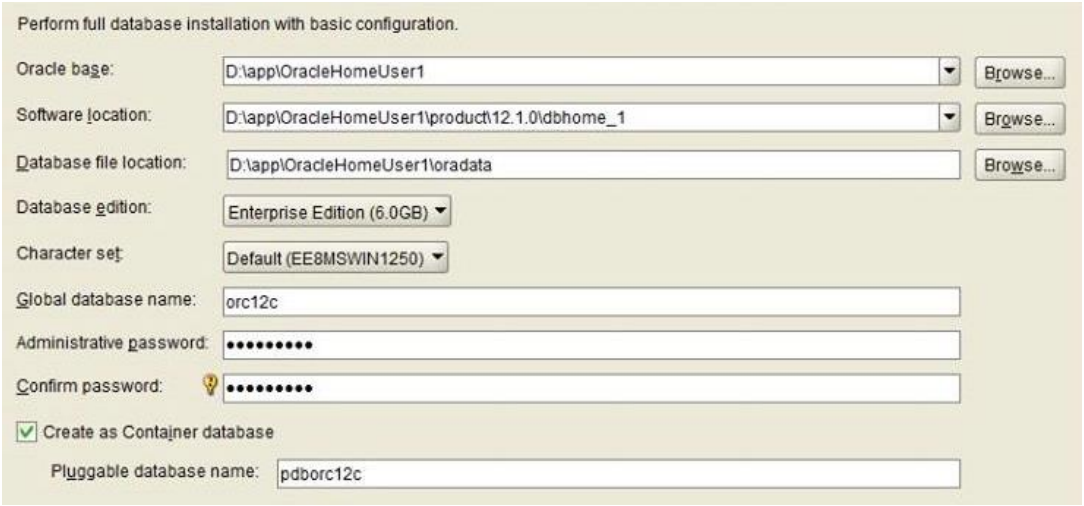
Potřebný software pro instalaci databáze Oracle 12c lze nalézt na webových stránkách společnosti Oracle v sekci Downloads.
Databázový systém je rozdělen do dvou archivovaných souborů. Aby bylo možné tyto dva soubory stáhnout z tohoto webu, je třeba zaškrtnout souhlas s licenčním ujednáním OTN a dále je třeba mít vytvořený bezplatný účet Oracle a být přihlášen. Jakmile jsou splněny tyto podmínky, je možné stáhnout oba soubory do počítače. Soubory lze stáhnout zdarma jen pro studijní a nekomerční využití.

K rozbalení těchto souborů je možné použít jakéhokoliv dekompresního nástroje. Je velmi důležité, aby oba soubory byly rozbaleny do jedné složky s libovolným názvem.

Pokud je vše v pořádku, lze přistoupit k samotné instalaci databáze Oracle 12c. Instalaci produktu se spustí poklepáním na položku s názvem „setup“. Na začátku instalace Oracle Database 12c je možné vyplnit své přihlašovací údaje k účtu Oracle, k dostávání nejnovějších informací o problémech při instalaci či konfiguraci tohoto produktu. V dalším kroku si lze zvolit, zda pokračovat se současnou verzí tohoto softwaru nebo jestli se má vyhledat nejnovější verze tohoto softwaru. Jelikož je tato současná verze produktu aktuální, lze tento krok přeskočit a vynechat tak vyhledávání aktualizací. V následujícím kroku si lze vybrat celkem ze tří způsobů instalace. Protože je potřeba vytvořit a nakonfigurovat databázi bude zvolena možnost „Create and configure a database“. V dalším kroku je možné vybrat typ stroje, na kterém bude databáze provozována. Zvolena bude možnost „Desktop“. Další krok se zabývá vytvořením tzv. domácího uživatele databáze (Oracle Home User). Tato možnost je dostupná teprve od verze 12.1. Pomocí tohoto uživatele lze spouštět databázové služby produktu Oracle Database 12c v prostředí Windows. Uživatel zvolí možnost „Create New Windows User“ a vytvoří například uživatele se jménem OracleHomeUser1 a nastaví mu heslo.

Obrázek 9: Ukázka vytvoření uživatele OracleHomeUser1

V pořadí již šestém kroku je možné zkontrolovat umístění databázového systému po instalaci. Tento krok také obnáší vytvoření databázové instance. Název databáze si uživatel zvolí jako orc12c a současně si nastaví heslo pro společné uživatele SYS a SYSTEM. Protože zde budou předvedeny základní práce se zásuvnými databázemi, je zde možné rovnou z této databázové instance udělat kontejnerovou databázi tím, že uživatel zaškrtne možnost „Create as Container database“. Obrázek č. 10 znázorňuje šestý krok instalace.



Obrázek 10: Ukázka vytvoření kontejnerové databáze orc12c

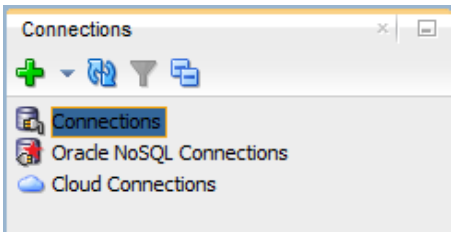
Jak je vidět, společně s kontejnerovou databází se vytvoří i první zásuvná databáze, která se bude jmenovat pdborc12c. Tato zásuvná databáze je defaultní a obsahuje ukázkové HR schéma. V dalším kroku už je jen shrnutí instalace a konfigurace. Nyní je možné pokračovat spuštěním samotné instalace produktu Oracle Database 12c, která obvykle trvá několik minut.

2.2.1 Ověření instalace

Pro ověření, zda instalace proběhla v pořádku, je možné například zkontrolovat, zda se vytvořily domovské složky pro domácího uživatele OracleHomeUser1. Domovskou složku uživatele OracleHomeUser1 lze nalézt na adrese „c:\app\jmenoDomovskehoUzivatele“. V tomto konkrétním případě je cesta ke složce „c:\app\OracleHomeUser1“. V této složce je podsložka s názvem „product“. Tato složka obsahuje nainstalovaný produkt databáze Oracle. Dále je zde také podsložka s názvem „oradata“, která obsahuje soubory vztahující se ke kontejnerové databázi orc12c. Dalším způsobem jak ověřit, zda je databáze nainstalována a spuštěna, je monitorovací nástroj v prostředí Windows s názvem „perfmon.exe“. Po spuštění tohoto nástroje by měl být v záložce „Paměť“ proces s názvem „oracle.exe“. Zde se lze také přesvědčit, jak velké množství fyzické paměti tento proces využívá.

2.2.2 Připojení k databázi

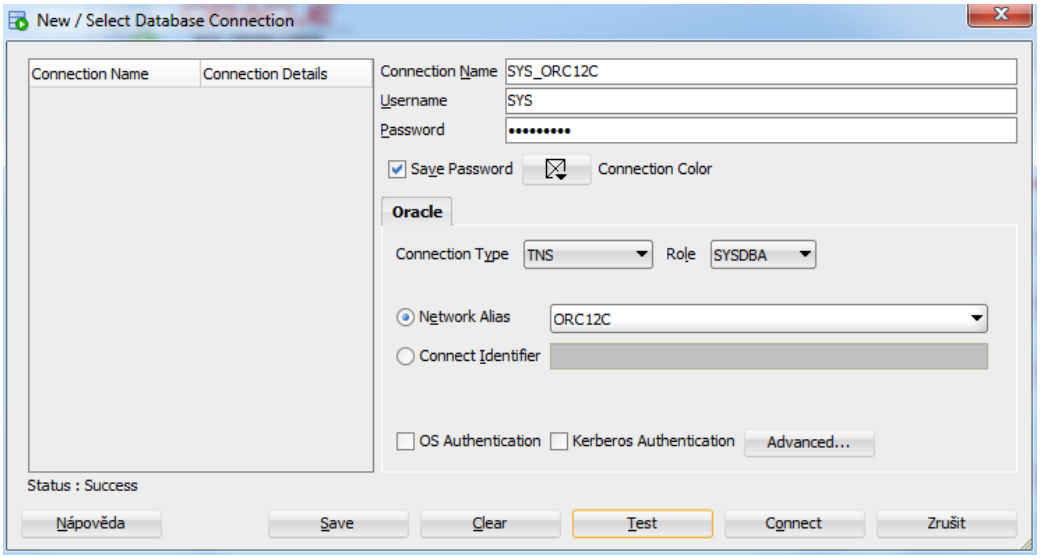
V předchozí kapitole bylo popsáno jak zkontrolovat, zda je databáze správně nainstalována a zda je spuštěna. Je-li databáze spuštěna, je možné se k ní připojit. Pro připojení ke kontejnerové databázi orc12c, která byla vytvořena a nakonfigurována v průběhu instalace produktu, bude využito nástroj SQL Developer. Nyní bude předvedeno jak se připojit k databázi orc12c s uživatelem SYS. Jakmile se spustí nástroj SQL Developer, na levé straně uživatel nalezne záložku „Connections“, kde označí položku se stejným názvem „Connections“ a kliknutím na zelené znaménko plus spustí dialog pro vytvoření nového připojení.



Obrázek 11: Záložka Connections v nástroji SQL Developer

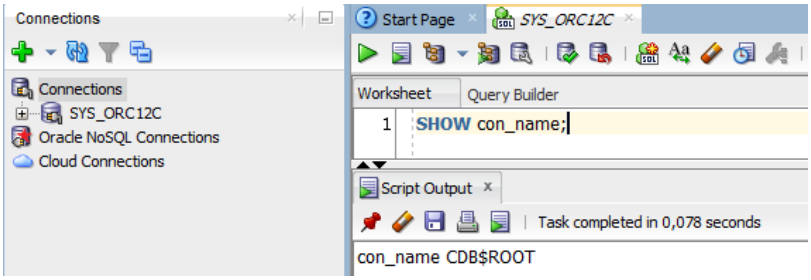
První co uživatel udělá je, že vybere vhodný název pro připojení k databázi. Název může být jakýkoliv, avšak protože v budoucnu zde může být vytvořeno mnoho podobných připojení, bude zvolen název „SYS_ORC12C“. V budoucnu, tento název uživateli napoví, o jaké připojení se jedná. Další věc, kterou je třeba udělat je zadat jméno a přístupové heslo uživatele, kterým se uživatel chce připojit. Bude zadáno jméno uživatele, tedy „SYS“ a jeho heslo. Nyní je potřeba zvolit typ připojení. Pokud bude zvolen základní typ připojení (ang. Basic), tak stačí jen správně vyplnit jméno databáze (SID), ke které se chce uživatel připojit. Hostitelské jméno a port jsou vyplněny automaticky. Dalším typem připojení, které bude v tomto tutoriálu použito, je TNS (Transparent Network Substrate). Tento typ připojení vyžaduje, aby databáze, ke které se bude uživatel připojovat, měla vytvořen tzv. síťový alias. Jak vytvořit síťový alias bude popsáno v kapitole 2.2.2.1. Uživatel vybere typ připojení TNS, a jako poslední věc, kterou musí udělat je vybrat uživatelskou roli. Protože se bude připojovat jako společný uživatel SYS, musí vybrat z nabídky roli SYSDBA.

Tlačítkem „Test“ lze vyzkoušet funkčnost připojení a v levém dolním rohu se zobrazí výsledek pokusu o spojení. Je-li vše v pořádku, je možné se připojit k databázi kliknutím na tlačítko „Connect“.



Obrázek 12: Ukázka vytvoření spojení s databází orc12c pomocí připojení TNS

Po vytvoření spojení se otevře pracovní list, do kterého lze zadávat SQL dotazy a komunikovat tak s databází. Pokud bude například zadán příkaz SHOW con_name, zobrazí se jméno databáze, se kterou je právě vedena komunikace. V případě, že se jedná o hlavní kontejner, tedy CDB, zobrazí se název CDB$ROOT.



Obrázek 13: Ukázka pracovního listu a zadání příkazu pro zobrazení názvu současného kontejneru

Připojit se k databázi lze i pomocí příkazové řádky systému Windows. Zadáním příkazu sqlplus / as sysdba se uživatel systému Windows přepne do dotazovacího režimu SQL a připojí se k databázi jako uživatel sysdba. Poté je možné opět odeslat příkaz SHOW con_name a jako výsledek se zobrazí opět CDB$ROOT.

2.2.2.1 Vytvoření síťového aliasu

Pokud uživatel potřebuje přistupovat k databázi pomocí proprietárního protokolu TNS, musí mít daná databáze vytvořen síťový alias. Pro vytvoření síťového aliasu je třeba upravit síťový konfigurační soubor s názvem „tnsname.ora“. (V tomto konkrétním případě je cesta k souboru následující: c:\app\OracleHomeUser1\product\12.1.0\dbhome_1\NETWORK\ADMIN).
Soubor lze otevřít pomocí jakéhokoliv textového editoru a nakonec souboru je potřeba vložit následující kousek kódu:

```
PDBORC12C =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = pdborc12c)
    )
  )
```

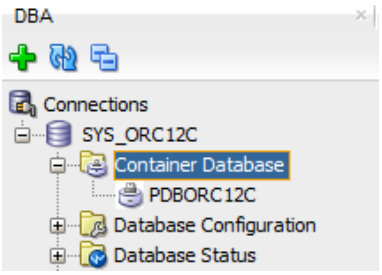
Obrázek 14: Ukázka vytvoření síťového aliasu pro službu pdborc12c

Na obrázku č. 14 je vytvářen síťový alias pro zásuvnou databázi PDBORC12C. První slovo je tedy název databáze, pro kterou je vytvářen síťový alias. Naopak poslední slovo je název síťové služby databáze, tedy pdborc12c. Po úpravě souboru, uživatel soubor uloží. Tímto byl vytvořen síťový alias pro databázi PDBORC12C. Síťový alias pro kontejnerovou databázi se vytváří automaticky při instalaci softwaru Oracle.

2.3 Základy práce se zásuvnými databázemi

V předchozím příkladu bylo pomocí nástroje SQL Developer vytvořeno připojení ke kontejnerové databázi se společným uživatelem SYS. Nyní zde bude předvedena práce se zásuvnými databázemi. Aby bylo možné pracovat v nástroji SQL Developer se zásuvnými databázemi, je třeba přidat spojení SYS_ORC12C do DBA navigátoru. Když to uživatel udělá, bude moci vykonávat kompletní administraci nad kontejnerovou databází i nad jednotlivými zásuvnými databázemi. Přidání spojení do DBA navigátoru je velmi jednoduché. Na levé straně SQL Developeru lze nalézt záložku s označením DBA. Uživatel opět klikne na zelený symbol plus, který mu umožní přidat nové spojení do DBA navigátoru. Z nabídky musí uživatel zvolit požadované spojení a následně kliknout na tlačítko přidat.

Pokud nyní uživatel rozklikne připojení SYS_ORC12C v DBA navigátoru a dále vybere například položku Container Database tak se mu zobrazí všechny momentálně připojené zásuvné databáze a lze s nimi nyní pracovat. Na obrázku č. 15 je vidět zatím jediná zásuvná databáze PDBORC12C, která byla vytvořena při instalaci databáze Oracle.

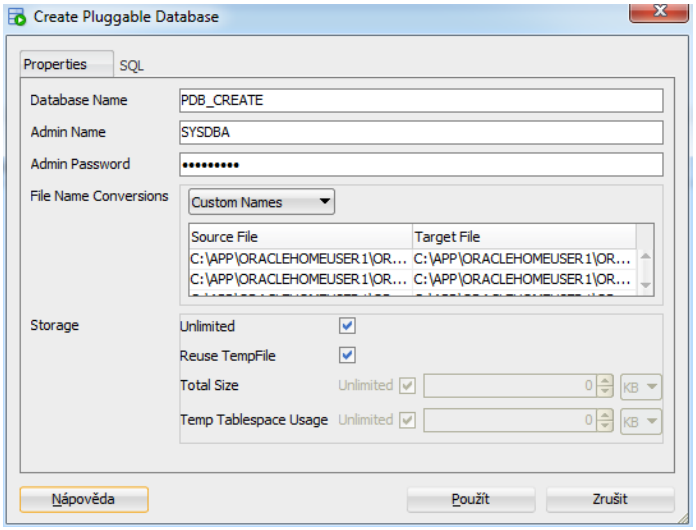


Obrázek 15: Záložka DBA a zásuvná databáze PDBORC12C s pomocí nástroje SQL Developer

2.3.1 Vytvoření PDB

Protože již uživatel získal plnou kontrolu nad administrací kontejnerové databáze orc12c, lze nyní ukázat jak vytvořit novou zásuvnou databázi. Existuje několik způsobů jak vytvořit novou zásuvnou databázi. Zásuvná databáze může vzniknout kopírováním. Nová zásuvná databáze vznikne kopírováním semínka PDB$SEED nebo kopírováním již existující databáze. V takovém případě se jedná o klonování zásuvné databáze. Dalším způsobem jak může vzniknout zásuvná databáze, je připojení databáze. Připojit lze databázi, která byla dříve odpojena z jiné či současné kontejnerové databáze.

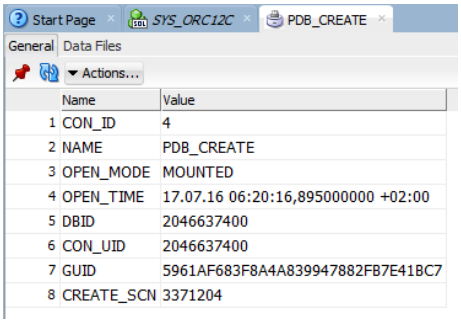
Nyní bude vytvořena zásuvná databáze kopírováním ze semínka. Pravým kliknutím na položku Container Database a vybráním možnosti Create Pluggable Database se otevře okno pro vytvoření nové PDB. Uživatel zadá název nové zásuvné databáze např. PDB_CREATE, jméno a heslo administrátora, tedy SYSDBA. Dále je třeba zvolit konverzi pro pojmenovávání souborů, uživatel zvolí obvyklé pojmenovávání, tedy volbu Custom Names. Po kliknutí na tlačítko použít, se do položky Container Database přidá nová zásuvná databáze s názvem PDB_CREATE.



Obrázek 16: Ukázka vytvoření nové PDB pomocí nástroje SQL Developer

2.3.1.1 Změna stavu databáze

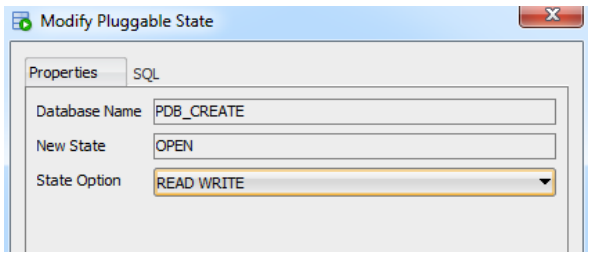
Stav databáze je velmi důležitým faktorem například při klonování databáze. Po vytvoření nové zásuvné databáze se tato databáze nachází ve stavu MOUNTED, jak je dobře vidět na obrázku č. 17.



Obrázek 17: Zobrazení současného stavu databáze PDB_CREATE (OPEN_MODE)

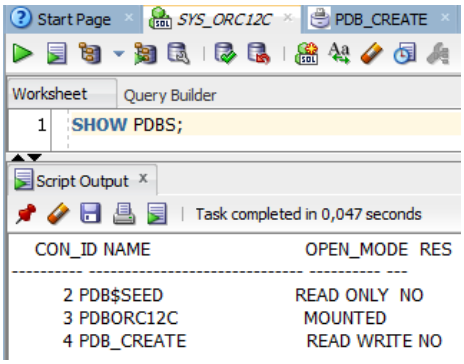
K tomu, aby bylo možné z databáze číst a zapisovat na ni data, je potřeba změnit její stav. V předchozí kapitole byla vytvořena nová zásuvná databáze PDB_CREATE. Tato databáze se nyní nachází ve stavu MOUNTED. Nyní se uživatel pokusí změnit její stav. Kliknutím pravým tlačítkem myši na PDB_CREATE a vybráním možnosti pro změnu stavu, tedy Modify State se otevře dialog pro změnu stavu databáze.

Nový stav databáze se automaticky nastaví na OPEN a uživatel si může vybrat, zda chce, aby byla databáze otevřena jen pro čtení, nebo i pro zápis. Uživatel vybere možnost READ WRITE a databáze tak přejde do stavu OPEN pro čtení i zápis.



Obrázek 18: Ukázka změny stavu databáze PDB_CREATE na OPEN READ WRITE

Nový stav databáze lze překontrolovat například příkazem SHOW PDBS, který zobrazí všechna PDB včetně jejich aktuálního stavu. Výsledek by měl vypadat následovně.

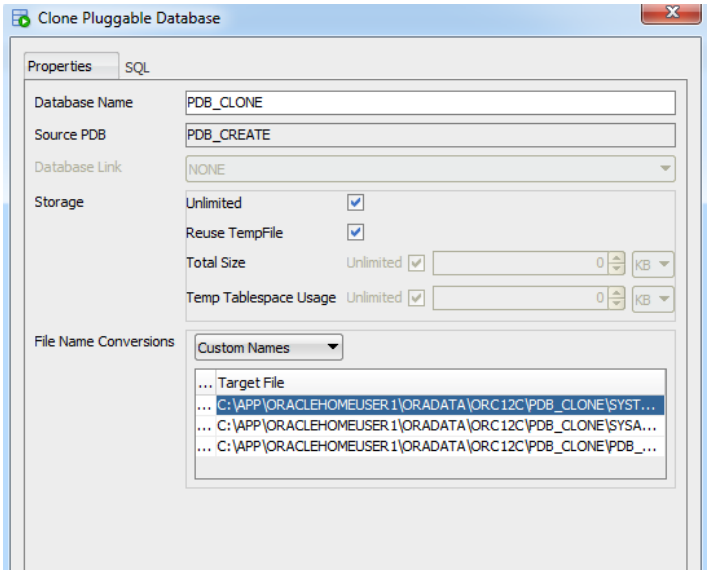


Obrázek 19: Nový stav databáze PDB_CREATE

2.3.2 Klonování PDB

Vytvoření nové zásuvné databáze lze docílit i klonováním databáze stávající. Proto, aby bylo možné nějakou stávající zásuvnou databázi klonovat, musí být tato databáze ve stavu READ ONLY. Nejprve je nutné změnit stav databáze PDB_CREATE na CLOSE a následně databázi znovu otevřít a nastavit stav na READ ONLY. Nyní je možné přistoupit k vytvoření klonu. Pravým tlačítkem myši uživatel klikne na databázi, kterou si přeje naklonovat a vybere možnost Clone Pluggable Database. Zobrazí se okno pro vytvoření klonu zásuvné databáze. Jako první je třeba zadat název nové databáze. Je zde také vidět zdrojová databáze, ze které se budou kopírovat soubory. Dále je třeba zvolit konverzi pro pojmenovávání souborů.

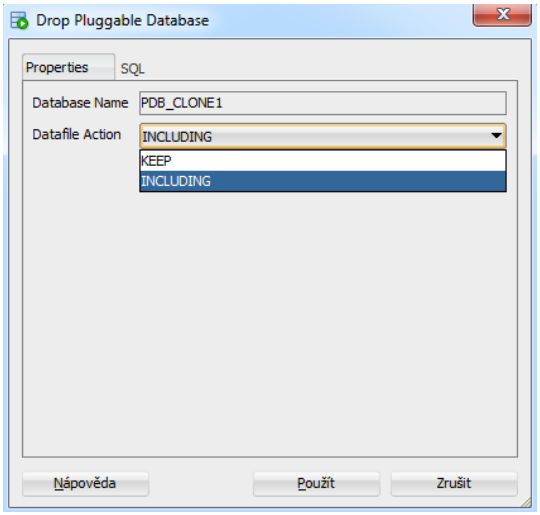
Uživatel opět vybere obvyklé pojmenovávání, tedy volbu Custom Names. Potvrzením se ze zdrojové databáze PDB_CREATE vytvoří nová zásuvná databáze PDB_CLONE.



Obrázek 20: Ukázka klonování databáze PDB_CREATE a vytvoření klonu PDB_CLONE

2.3.3 Smazání PDB

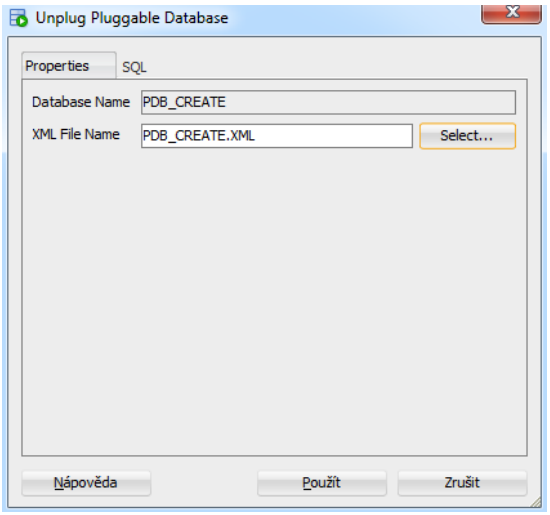
V případě, že se s danou zásuvnou databázi již nebude pracovat, může být tato databáze odstraněna z kontejnerové databáze. Před smazáním zásuvné databáze je dobré zvážit, zda si ponechat přidružené datové soubory nebo nikoliv. Při odstraňování musí být daná PDB ve stavu CLOSE. Pravým tlačítkem na požadovanou databázi a výběrem Drop Pluggable Database se otevře okno pro odstranění databáze. Pokud se uživatel rozhodne, že si ponechá přidružené datové soubory tak musí zvolit možnost KEEP, jinak vybere možnost INCLUDING a dojde k odstranění zásuvné databáze včetně jejich datových souborů.



Obrázek 21: Ukázka vymazání databáze PDB_CLONE1

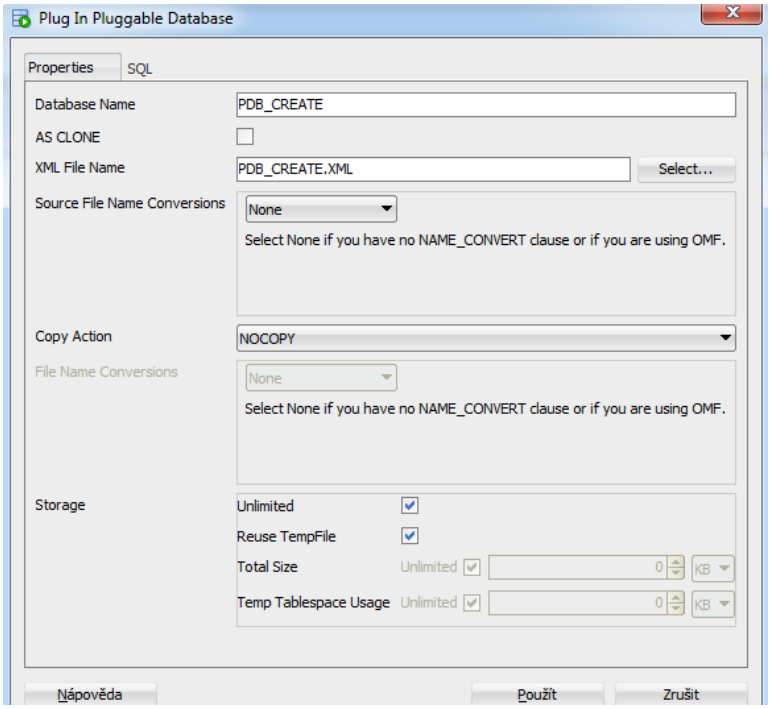
2.3.4 Připojení a odpojení PDB

Odpojení zásuvné databáze se využívá při přesunutí databáze do jiné kontejnerové databáze, či pokud se s databází delší dobu nepracuje. Než dojde k samotnému odpojení databáze, je třeba tuto databázi převést do stavu CLOSE. Jakmile je databáze odpojena, nachází se ve stavu MOUNTED. Pro odpojení zásuvné databáze je třeba kliknout na vybranou PDB a zvolit možnost Unplug Pluggable Database. Do okna, které se objeví, je třeba vyplnit pouze jméno XML souboru, který bude uchovávat umístění datových souborů této odpojené databáze pro případné příští zapojení.



Obrázek 22: Ukázka odpojení databáze PDB_CREATE

Odpojená databáze PDB_CREATE by za normálních podmínek byla využita v jiné kontejnerové databázi, avšak pro ukázku připojení zásuvné databáze bude předvedeno, jak tuto databázi připojit zpět do kontejnerové databáze ORC12C. Pravým tlačítkem myši uživatel klikne na položku Container Database a zvolí možnost Plug In Pluggable Database. Zadá název databáze, kterou si přeje připojit a vybere správný XML soubor, který obsahuje informace o umístění datových souborů této PDB. Kliknutím na tlačítko použít, se připojí databáze PDB_CREATE zpět do CDB.

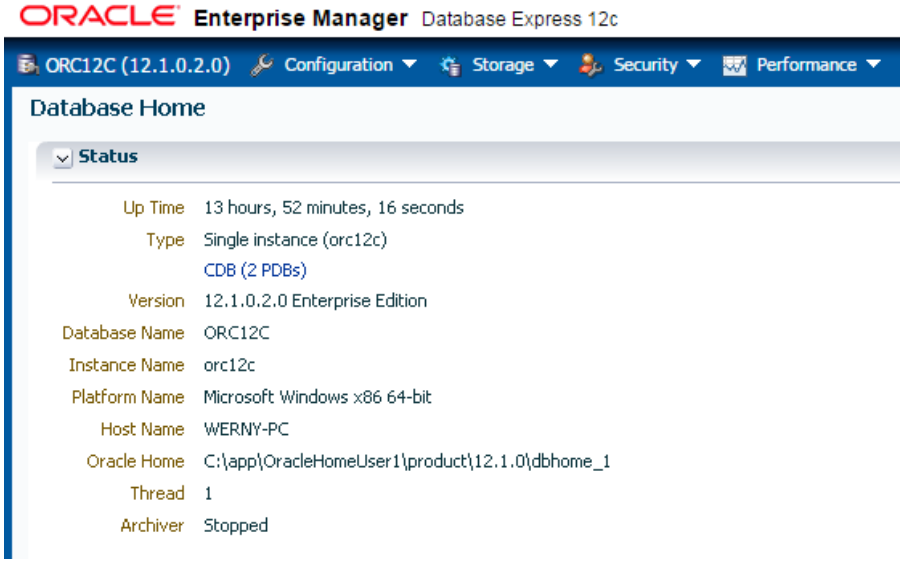


Obrázek 23: Ukázka připojení databáze PDB_CREATE

2.3.5 Zásuvné databáze v prostředí EM Database Express

Oracle Enterprise Manager Database Express nebo zkráceně EM Database Express je webové rozhraní pro základní správu a monitoring Oracle Database. Do tohoto nástroje se lze dostat pomocí jakéhokoliv internetového prohlížeče. Po zadání URL adresy ve tvaru https://<hostname>:<port>/em se zobrazí přihlašovací formulář pro přístup do tohoto nástroje. V tomto konkrétním případě vypadá URL adresa takto https://localhost:5500/em. Číslo portu, na kterém bude vedena komunikace s tímto nástrojem, se zobrazuje v posledním kroku instalace Oracle Database.

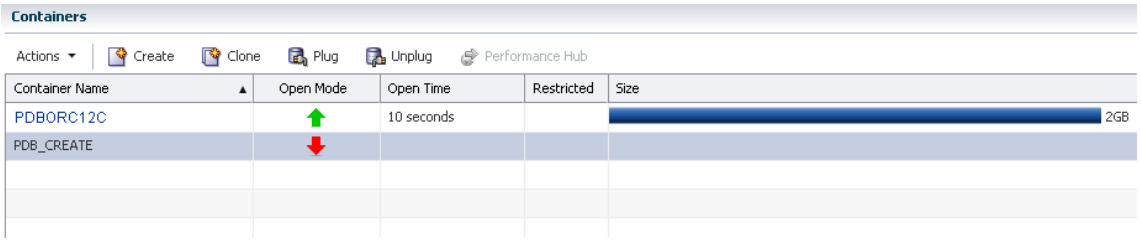
Jiný způsob jak zjistit číslo portu bude ukázán v bonusových příkladech na konci praktické části této práce. Aby bylo možné spravovat databázi, musí se uživatel přihlásit jako společný uživatel SYS. Jakmile se přihlásí, zobrazí se mu základní informace o kontejnerové databázi ORC12C. V levém horním rohu si lze například všimnout, jak dlouho je databázová instance spuštěna či kolik zásuvných databází tato kontejnerová databáze obsahuje.



Obrázek 24: Informace o databázové instanci orc12c s pomocí nástroje EM Database Express

Pokud budou uživatele zajímat bližší informace o zásuvných databázích, kliknutím na zvýrazněný text CDB (2 PDBs) se mu otevře nová stránka s informacemi o daných PDBs. Na tomto místě lze provádět stejné operace se zásuvnými databázemi, stejně jako tomu bylo pomocí nástroje SQL Developer. Lze tedy vytvářet, klonovat či mazat PDB databáze a samozřejmě měnit jejich stavy.

Na obrázku č. 25 dole jsou vidět zásuvné databáze PDBORC12C a PDB_CREATE. Zelená šipka ve sloupci s názvem „Open Mode“ znázorňuje otevřený stav databáze. Databáze PDBORC12C je připravena pro čtení i zápis. Naproti tomu červená šipka signalizuje, že je databáze uzavřená.



Obrázek 25: Zobrazení zásuvných databází s pomocí nástroje EM Database Express

2.4 Správa paměti databáze Oracle 12c

V této kapitole budou s pomocí nástroje SQL*Plus předvedeny základní úkony prováděné při správě paměti databáze Oracle 12c. Například zde bude předvedeno, kde zjistit kolik paměti zabírají různé komponenty SGA paměti, jak docílit zvýšení SGA paměti změnou inicializačního parametru SGA_MAX_SIZE či jak povolit funkci In-Memory. Uživatel si spustí nástroj SQL*Plus a připojí se ke kontejnerové databázi ORC12C příkazem „sqlplus / as sysdba“. Kompletní informace o globální systémové oblasti paměti (SGA) si uživatel zobrazí příkazem SELECT * FROM V$SGAINFO. Na obrázku č. 26 jsou vidět všechny komponenty SGA paměti včetně velikosti paměti, kterou zabírají. Také je zde zobrazena velikost zrníčka (Granule Size), o které byla řeč v kapitole 1.2.2.1.

```
SQL> SELECT * FROM V$SGAINFO;

NAME                                 BYTES RES     CON_ID
-------------------------------- ---------- --- ----------
Fixed SGA Size                     3048872 No           0
Redo Buffers                      13725696 No           0
Buffer Cache Size               1962934272 Yes          0
In-Memory Area Size                      0 No           0
Shared Pool Size                 536870912 Yes          0
Large Pool Size                   16777216 Yes          0
Java Pool Size                    16777216 Yes          0
Streams Pool Size                        0 Yes          0
Shared IO Pool Size              117440512 Yes          0
Data Transfer Cache Size                 0 Yes          0
Granule Size                      16777216 No           0

NAME                                 BYTES RES     CON_ID
-------------------------------- ---------- --- ----------
Maximum SGA Size                2550136832 No           0
Startup overhead in Shared Pool  164926096 No           0
Free SGA Memory Available                0              0

14 rows selected.
```

Obrázek 26: Ukázka zobrazení informací o SGA paměti prostřednictvím nástroje SQL*Plus

Z tohoto výpisu je také patrná velikost paměti, kterou zabírá oblast In-Memory. Velikost je defaultně nastavena na 0 bajtů. Je to z toho důvodu, že funkce In-Memory je po instalaci Oracle Database 12c defaultně deaktivována. Jak aktivovat oblast In-Memory bude předvedeno v kapitole 2.4.3. Další asi nejzajímavější informaci, kterou lze získat z tohoto výpisu je maximální velikost SGA paměti (Maximum SGA Size). Tato velikost je uložena v inicializačním parametru SGA_MAX_SIZE. V teoretické části práce bylo zmíněno, že hodnota nastavena v tomto parametru nemůže být nikdy překročena. Pro méně podrobné informace lze použít příkaz SHOW SGA.

2.4.1 Automatická správa paměti

Oracle databáze umožňují automatickou správu paměti. Nastavením parametru SGA_TARGET na jinou než nulovou hodnotu, dojde ke spuštění funkce automatické správy paměti. Hodnota tohoto parametru musí být nastavena stejně nebo méně než hodnota parametru SGA_MAX_SIZE. Tento parametr nahrazuje jiné parametry, které kontrolují naalokovanou paměť jednotlivých komponentů a dynamicky mění jejich velikosti podle toho, jak je zrovna potřeba. Funkce automatické správy paměti je defaultně spuštěná a hodnota parametru SGA_TARGET má stejnou hodnotu jako je hodnota parametru SGA_MAX_SIZE. Na rozdíl od parametru SGA_MAX_SIZE, hodnotu parametru SGA_TARGET lze měnit i za běhu databázové instance a změna se projeví okamžitě bez nutnosti restartovat databázovou instanci. Pro deaktivování automatické správy paměti stačí nastavit parametr SGA_TARGET na nulovou hodnotu. Příkazem SHOW PARAMETER SGA lze zobrazit parametry, v jejichž názvu je slovo SGA. Jak je vidět, jsou zde kromě jiných parametrů, jak parametr SGA_MAX_SIZE s hodnotou 2432 MB, tak i parametr SGA_TARGET, který má nastavenou stejnou hodnotu. Pokud se nyní uživatel pokusí změnit velikost parametru SGA_TARGET na větší hodnotu, například 3000 MB, na obrázku č. 27 je vidět, co se stane. Změnu parametru SGA_TARGET lze provést příkazem ALTER SYSTEM SET SGA_TARGET=3000M SCOPE=BOTH. Klauzule SCOPE v tomto příkazu představuje místo, kde má být provedena úprava tohoto parametru. V tomto konkrétním případě byla zvolena možnost BOTH, což znamená, že změna bude provedena jak v souboru s inicializačními parametry s názvem SPFILE, který je uložen na pevném disku, tak i v paměti.

Pokud by uživatel zvolil možnost MEMORY, změna se projeví okamžitě a přetrvá, dokud nebude ukončena databázová instance. Zatímco pokud by uživatel vybral možnost SPFILE, změnila by se hodnota parametru přímo v souboru SPFILE a tato změna by se stala trvalou. K tomu by však navíc ještě musel uživatel restartovat databázovou instanci.

```
SQL> SHOW PARAMETER SGA;

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
lock_sga                             boolean     FALSE
pre_page_sga                         boolean     TRUE
sga_max_size                         big integer 2432M
sga_target                           big integer 2432M
unified_audit_sga_queue_size         integer     1048576
SQL>
SQL>
SQL> ALTER SYSTEM SET SGA_TARGET=3000M SCOPE=BOTH;
ALTER SYSTEM SET SGA_TARGET=3000M SCOPE=BOTH
*
ERROR at line 1:
ORA-02097: parameter cannot be modified because specified value is invalid
ORA-00823: Specified value of sga_target greater than sga_max_size
```

Obrázek 27: Neúspěšná změna parametru SGA_TARGET

Z obrázku č. 27 je poznat, že se úprava parametru nezdařila. Je to z toho důvodu, že hodnota parametru SGA_TARGET nesmí být nikdy větší než hodnota parametru SGA_MAX_SIZE, a jelikož požadovaná hodnota 3000 MB je větší než hodnota 2432 MB, nelze tuto změnu provést. Příkazem SHOW PARAMETER SPFILE lze zjistit umístění souboru s inicializačními parametry.

```
SQL> SHOW PARAMETER SPFILE;

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
spfile                               string      C:\APP\ORACLEHOMEUSER1\PRODUCT
                                                 \12.1.0\DBHOME_1\DATABASE\SPFI
                                                 LEORC12C.ORA
SQL>
```

Obrázek 28: Ukázka zobrazení umístění souboru s inicializačními parametry

Protože mají parametry SGA_MAX_SIZE a SGA_TARGET nastavenu stejnou hodnotu, nelze hodnotu parametru SGA_TARGET zvýšit, ale naopak ji lze snížit. Zadáním příkazu ALTER SYSTEM SET SGA_TARGET=2400M SCOPE=BOTH uživatel zajistí snížení hodnoty parametru SGA_TARGET na hodnotu 2400 MB a změny se projeví jak v paměti, tak i v inicializačním souboru spfileorc12c. Opětovným zadáním příkazu SHOW PARAMETER SGA lze překontrolovat výsledek.

```
SQL> ALTER SYSTEM SET SGA_TARGET=2400M SCOPE=BOTH;

System altered.

SQL> SHOW PARAMETER SGA;

NAME                                 TYPE        VALUE
------------------------------------ ----------- --------
lock_sga                             boolean     FALSE
pre_page_sga                         boolean     TRUE
sga_max_size                         big integer 2432M
sga_target                           big integer 2400M
unified_audit_sga_queue_size         integer     1048576
SQL>
```

Obrázek 29: Úspěšná změna parametru SGA_TARGET

Změnou parametru SGA_TARGET dojde k automatickému přerozdělení paměti mezi jednotlivé komponenty. V případech jako je tento, kdy byla nastavena hodnota parametru SGA_TARGET na menší hodnotu než je hodnota inicializačního parametru SGA_MAX_SIZE, může vzniknout část volné paměti. Proto je vhodné nastavit hodnotu parametru SGA_TARGET na stejnou hodnotu jako je hodnota SGA_MAX_SIZE. Bude tak využita veškerá dostupná paměť, která je k dispozici.

2.4.2 Změna inicializačního parametru SGA_MAX_SIZE

Stejně jako lze měnit hodnotu parametru SGA_TARGET, lze měnit i velikost parametru SGA_MAX_SIZE, který udává velikost SGA paměti. Protože jde o inicializační parametr (parametr potřebný při inicializaci databázové instance), není možné tento parametr měnit za běhu databázové instance. Pokud bychom chtěli změnit velikost tohoto parametru za běhu databázové instance, dospěli bychom k následujícím chybám. Příkazy byly odeslány z nástroje SQL Developer.

```
Error starting at line : 1 in command -
ALTER SYSTEM SET SGA_MAX_SIZE=3000M SCOPE=MEMORY
Error report -
SQL Error: ORA-02095: Uvedený inicializační parametr nemůže být měněn
02095. 00000 -  "specified initialization parameter cannot be modified"
*Cause:    The specified initialization parameter is not modifiable
Error starting at line : 1 in command -
ALTER SYSTEM SET SGA_MAX_SIZE=3000M SCOPE=BOTH
Error report -
SQL Error: ORA-02095: Uvedený inicializační parametr nemůže být měněn
02095. 00000 -  "specified initialization parameter cannot be modified"
*Cause:    The specified initialization parameter is not modifiable
```

Obrázek 30: Neúspěšné pokusy o změnu inicializačního parametru SGA_MAX_SIZE

Příkaz pro provedení změny velikosti tohoto parametru sice lze provést za běhu databázové instance, avšak změna bude provedena až po vypnutí a opětovném zapnutí databáze. Jediný funkční příkaz, který změní parametr SGA_MAX_SIZE vypadá takto ALTER SYSTEM SET SGA_MAX_SIZE=3000M SCOPE=SPFILE. Inicializační parametr tak bude změněn přímo v souboru spfileorc12c.ora. Jakmile dojde k opětovnému spuštění databázové instance, může se uživatel přesvědčit, zda byla změna hodnoty parametru provedena.

```
SQL> SHOW PARAMETER SGA;

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------
lock_sga                             boolean     FALSE
pre_page_sga                         boolean     TRUE
sga_max_size                         big integer 3008M
sga_target                           big integer 2400M
unified_audit_sga_queue_size         integer     1048576
SQL>
```

Obrázek 31: Úspěšná změna parametru SGA_MAX_SIZE a kontrola SGA

2.4.3 Paměťová oblast In-Memory

Nová oblast paměti s názvem In-Memory Column Store (IMCS) je defaultně deaktivována. Pro aktivování této technologie je třeba nastavit inicializační parametr INMEMORY_SIZE na jinou než nulovou hodnotu. Velikost tohoto parametru záleží na celkové velikosti paměti, kterou budou používat databázové objekty označené klíčovým slovem INMEMORY.

Databázové objekty s tímto označením budou ukládány do komponenty IMCS. Velikost IMCS by měla být o něco vyšší, pro případný růst těchto databázových objektů. Jakmile bude odhadnuta patřičná velikost pro IMCS, lze přistoupit k úpravě parametru INMEMORY_SIZE. Tento parametr je stejně jako parametr SGA_MAX_SIZE parametrem inicializačním. Změna parametru se tedy projeví až po vypnutí a opětovném spuštění databázové instance. Obrázek č. 32 ukazuje příkaz ALTER SYSTEM SET INMEMORY_SIZE=500M SCOPE=SPFILE, který nastaví hodnotu parametru na 500 MB a změny uloží do inicializačního souboru.

```
SQL> ALTER SYSTEM SET INMEMORY_SIZE = 500M SCOPE=SPFILE;

System altered.

SQL> SHUTDOWN;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
SQL> STARTUP;
ORACLE instance started.

Total System Global Area 3154116608 bytes
Fixed Size                  3050600 bytes
Variable Size            1224737688 bytes
Database Buffers         1375731712 bytes
Redo Buffers               13725696 bytes
In-Memory Area            536870912 bytes
Database mounted.
Database opened.
SQL>
```

Obrázek 32: Změna parametru INMEMORY_SIZE na 500MB

Jak je z obrázku poznat, nastavení inicializačního parametru INMEMORY_SIZE proběhlo úspěšně. Po restartování databázové instance je vidět, že oblast In-Memory již neobsahuje nulovou hodnotu, což znamená, že je aktivní. V multitenantním prostředí lze nastavit tento parametr pro každou PDB zvlášť. Pro nastavení tohoto parametru v zásuvné databázi je třeba se pohybovat v konkrétní PDB s patřičným uživatelským oprávněním.

2.4.3.1 Deaktivace In-Memory

Pro deaktivaci funkce In-Memory stačí resetovat inicializační parametr INMEMORY_SIZE. Resetování parametru lze provést příkazem na obrázku č. 33. Po restartu databázové instance je vidět, že hodnota tohoto parametru byla změněna na původní hodnotu a funkce In-Memory byla tímto deaktivována.

```
SQL> ALTER SYSTEM RESET INMEMORY_SIZE SCOPE=SPFILE;

System altered.

SQL> SHUTDOWN IMMEDIATE;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP;
ORACLE instance started.

Total System Global Area 3154116608 bytes
Fixed Size                  3050600 bytes
Variable Size            1224737688 bytes
Database Buffers         1912602624 bytes
Redo Buffers               13725696 bytes
Database mounted.
Database opened.
SQL> SHOW PARAMETER INMEMORY_SIZE;

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------
inmemory_size                        big integer 0
```

Obrázek 33: Ukázka resetování parametru INMEMORY_SIZE

2.4.4 Přiřazení databázového objektu do In-Memory

Náležitosti, jež je třeba vykonat pro aktivaci technologie In-Memory již byly popsány v předchozí kapitole. Nyní už zbývá jen zvolit databázové objekty, které mají být převedeny do sloupcového formátu. Vybrané databázové objekty musí být patřičně označeny proto, aby databázový systém věděl, že tyto objekty nebude posílat klasicky do buffer cache, ale bude je posílat do komponenty IMCS.

Oracle poskytuje ke každé verzi Oracle Database tzv. ukázková schémata, která je možné si doinstalovat do libovolné databáze. Tyto schémata jsou založena na fikci ukázkové společnosti, která prodává určitý druh zboží. Jedná se o mezinárodní společnost, která má obchodní zástupce hned v několika zemích. Společnost se skládá z několika sekcí, kde je každá z těchto sekcí reprezentována ukázkovým databázovým schématem.

Jedním z těchto databázových schémat je schéma s názvem SH (Sales History). Toto schéma je navrženo pro ukázky efektivní práce s objemnými daty. Na tomto schématu bude předvedena práce s In-Memory. Jak toto databázové schéma získat bude popsáno v jednom s dodatečných příkladů na konci praktické části této práce. S předpokladem, že je vytvořena nejlépe nová zásuvná databáze, na ní nainstalované databázové schéma SH a odemknutý uživatelský účet SH, je možné spustit nástroj SQL*Plus, ve kterém bude předvedeno jak pracovat s technologií In-Memory.

Nejprve by se měl uživatel přesvědčit, jaké je nastavení inicializačních parametrů pro aktivování In-Memory. Dále je třeba zkontrolovat, zda je zásuvná databáze, v tomto konkrétním případě PDB_TEST, otevřená a připravena pro čtení a zápis. Pokud je vše v pořádku může se uživatel přihlásit do zásuvné databáze PDB_TEST s uživatelským účtem SH, který je taktéž součástí ukázkových schémat. SQL dotazem SELECT table_name FROM user_tables uživatel zjistí, které tabulky mu jsou k dispozici pro práci s In-Memory. Uživatel vybere například tabulku s názvem SALES, která má nejvíce záznamů. Při vytváření tabulky či jiného databázového objektu, lze aktivovat funkci In-Memory tak, že na konec příkazu pro vytvoření objektu se přidá klauzule INMEMORY. Jelikož tabulka SALES již existuje, musí ji uživatel upravit příkazem ALTER TABLE SALES INMEMORY, tak bude u tabulky SALES aktivována funkce INMEMORY.

```
SQL> conn sh/sh@pdb_test;
Connected.
SQL>
SQL> SELECT COUNT(*) FROM SALES;

  COUNT(*)
----------
    918843

SQL> ALTER TABLE SALES INMEMORY;

Table altered.
```

Obrázek 34: Ukázka připojení uživatele SH, výpočet záznamů v tabulce SALES a označení tabulky SALES klauzulí INMEMORY

Nyní je zajištěno, že tabulka SALES bude zpracována komponentou IMCS. Pokud by uživatel chtěl vybrat jen některé sloupce tabulky nebo naopak některé sloupce vyloučit z In-Memory lze použít dotazu ALTER TABLE SALES INMEMORY NO INMEMORY(prod_id) a tím bude sloupec s názvem prod_id vyloučen z In-Memory. Pro porovnání budou vybrány ještě další dvě tabulky a budou také nastaveny pro zpracování IMCS. Avšak v tomto případě bude zvolena i konkrétní kompresní úroveň a úroveň priority. Příkazem na následujícím obrázku č. 35 lze zobrazit současné nastavení tabulek z hlediska funkce In-Memory. Je zde také vidět, jak byly upraveny tabulky PRODUCTS a CUSTOMERS.

```
SQL> ALTER TABLE PRODUCTS INMEMORY PRIORITY HIGH MEMCOMPRESS FOR QUERY LOW;

Table altered.

SQL> ALTER TABLE CUSTOMERS INMEMORY PRIORITY CRITICAL MEMCOMPRESS FOR QUERY HIGH;

Table altered.

SQL> SELECT TABLE_NAME, INMEMORY, INMEMORY_PRIORITY, INMEMORY_COMPRESSION
  2  FROM user_tables WHERE TABLE_NAME IN('CUSTOMERS','SALES','PRODUCTS');

TABLE_NAME           INMEMORY INMEMORY INMEMORY_COMPRESS
-------------------- -------- -------- -----------------
PRODUCTS             ENABLED  HIGH     FOR QUERY LOW
CUSTOMERS            ENABLED  CRITICAL FOR QUERY HIGH
SALES
```

Obrázek 35: Ukázka nastavení kompresní úrovně a úrovně priority

Tabulka CUSTOMERS byla nastavena na kritický stupeň priority, což znamená, že tato tabulka bude naplněna do IMCS jako první. Další tabulkou, která je adeptem pro naplnění do IMCS je tabulka PRODUCTS. O tabulce SALES se z tohoto výpisu nedá příliš zjistit. Je to z toho důvodu, že se jedná o tzv. Partition table. Rozsáhlé tabulky bývají rozděleny do menších lépe kontrolovatelných oddílů, které jsou při dotazování rychleji přístupné, protože není potřeba skenovat celou tabulku. Další obrázek č. 36 ukazuje současný obsah komponenty IMCS. Výpis lze získat příkazem na obrázku, který zobrazí specifikované informace z materializovaného pohledu V$IM_SEGMENTS.

```
SQL> select segment_name, populate_status, inmemory_priority, inmemory_compression
  2  from v$im_segments;

SEGMENT_NAME         POPULATE_ INMEMORY INMEMORY_COMPRESS
-------------------- --------- -------- -----------------
CUSTOMERS            COMPLETED CRITICAL FOR QUERY HIGH

SQL>
```

Obrázek 36: Ukázka zobrazení obsahu IMCS prostřednictvím materializovaného pohledu V$IM_SEGMENTS

Pro výpis byl vybrán název segmentu, stav naplnění, stupeň priority a kompresní úroveň. Lze si však také vypsat velikost paměti, kterou daný segment zabírá v IMCS přidáním sloupce s názvem INMEMORY_SIZE nebo pokud by uživatele zajímala velikost na disku počítače, kterou tento segment zabírá, stačí přidat sloupec s názvem BYTES. V tomto konkrétním případě je tabulka CUSTOMERS naplněna do IMCS. Tento databázový objekt má totiž nastavenou úroveň priority na kritickou a stejně tak kompresní poměr je nastaven na nejvyšší možnou úroveň. V IMCS jsou udržována jen kritická data, která jsou označena klauzulí INMEMORY. To, že byly označeny ještě další tabulky, aby byly zpracovány IMCS neznamená, že se musí do IMCS naplnit ihned. Pokud nemají data nastavenu kritickou úroveň priority a vysokou kompresní úroveň, jsou tyto tabulky naplněny do IMCS až tehdy, je-li třeba s nimi pracovat. Obsah IMCS se mění na základě toho, se kterými databázovými objekty se zrovna pracuje.

Co je třeba mít také na paměti je to, že naplňování databázových objektů do IMCS, zvláště pak v případě opravdu rozsáhlých databázových objektů může nějakou dobu trvat.

2.4.5 Buffer cache vs. In-Memory

Na posledním příkladu, který se týká technologie In-Memory bude předvedeno srovnání rychlosti vykonání SQL dotazu pomocí komponenty Buffer cache a In-Memory Column Store. Bude využito databázového ukázkového schématu SH. Na obě komponenty SGA paměti bude odeslán stejný SQL dotaz a pro zpřesnění výsledků bude na každou z komponent tento dotaz odeslán dvakrát.

Po odeslání SQL dotazu uživatel požaduje, aby se vrátil celkový zisk za prodej produktu „Deluxe Mouse“, jako filtr bude zvolen den v týdnu (Monday) a pohlaví zákazníka (F). Příkazem „set timing on“ se aktivuje zobrazení uplynulého času při provádění SQL dotazů.

Uživatel se přihlásí jako uživatel SH k databázi PDB_TEST, která musí být ve stavu OPEN READ WRITE a odešle SQL dotaz. Po navrácení výsledků se na konci výpisu zobrazí uplynulý čas. První pokus o provedení dotazu zabral 1.61 vteřin. Po provedení druhého pokusu je vidět, že se čas na provedení dotazu zvýšil na 2.95 vteřin. Pokusy jsou zaznamenány na obrázku č. 37.

```
SQL> select p.prod_name, c.cust_gender, t.day_name, SUM(s.amount_sold) AS TotalSold
  2  FROM sh.sales s, sh.products p, sh.customers c, sh.times t
  3  WHERE p.prod_id = s.prod_id AND c.cust_id = s.cust_id
  4  AND t.time_id = s.time_id AND p.prod_name = 'Deluxe Mouse'
  5  AND c.cust_gender = 'F' AND t.day_number_in_week = 1
  6  GROUP BY p.prod_name, c.cust_gender, t.day_name;

PROD_NAME                                          C DAY_NAME   TOTALSOLD
-------------------------------------------------- - --------- ----------
Deluxe Mouse                                       F Monday      19053.35

Elapsed: 00:00:01.61
SQL> select p.prod_name, c.cust_gender, t.day_name, SUM(s.amount_sold) AS TotalSold
  2  FROM sh.sales s, sh.products p, sh.customers c, sh.times t
  3  WHERE p.prod_id = s.prod_id AND c.cust_id = s.cust_id
  4  AND t.time_id = s.time_id AND p.prod_name = 'Deluxe Mouse'
  5  AND c.cust_gender = 'F' AND t.day_number_in_week = 1
  6  GROUP BY p.prod_name, c.cust_gender, t.day_name;

PROD_NAME                                          C DAY_NAME   TOTALSOLD
-------------------------------------------------- - --------- ----------
Deluxe Mouse                                       F Monday      19053.35

Elapsed: 00:00:02.95
```

Obrázek 37: Zpracování SQL dotazu pomocí Buffer Cache

Pro zpracování SQL dotazu komponentou IMCS je důležité, databázový objekt označit klauzulí INMEMORY. Aby byl rozdíl měření co největší, bude nastavena i úroveň priority a kompresní stupeň. Po označení tabulky SALES klauzulí INMEMORY se data v řádkovém formátu přesunou do komponenty In-Memory Column Store, kde budou převedeny do sloupcového formátu, následuje komprese těchto dat, která se provádí speciálními kompresními algoritmy. Data jsou nahrána do IMCS téměř okamžitě.

```
SQL> ALTER TABLE SALES INMEMORY PRIORITY HIGH MEMCOMPRESS FOR QUERY HIGH;

Table altered.

Elapsed: 00:00:00.02
SQL> select p.prod_name, c.cust_gender, t.day_name, SUM(s.amount_sold) AS TotalSold
  2  FROM sh.sales s, sh.products p, sh.customers c, sh.times t
  3  WHERE p.prod_id = s.prod_id AND c.cust_id = s.cust_id
  4  AND t.time_id = s.time_id AND p.prod_name = 'Deluxe Mouse'
  5  AND c.cust_gender = 'F' AND t.day_number_in_week = 1
  6  GROUP BY p.prod_name, c.cust_gender, t.day_name;

PROD_NAME                                          C DAY_NAME   TOTALSOLD
-------------------------------------------------- - --------- ----------
Deluxe Mouse                                       F Monday      19053.35

Elapsed: 00:00:00.07
SQL> select p.prod_name, c.cust_gender, t.day_name, SUM(s.amount_sold) AS TotalSold
  2  FROM sh.sales s, sh.products p, sh.customers c, sh.times t
  3  WHERE p.prod_id = s.prod_id AND c.cust_id = s.cust_id
  4  AND t.time_id = s.time_id AND p.prod_name = 'Deluxe Mouse'
  5  AND c.cust_gender = 'F' AND t.day_number_in_week = 1
  6  GROUP BY p.prod_name, c.cust_gender, t.day_name;

PROD_NAME                                          C DAY_NAME   TOTALSOLD
-------------------------------------------------- - --------- ----------
Deluxe Mouse                                       F Monday      19053.35

Elapsed: 00:00:00.07
```

Obrázek 38: Zpracování SQL dotazu pomocí komponenty IMCS

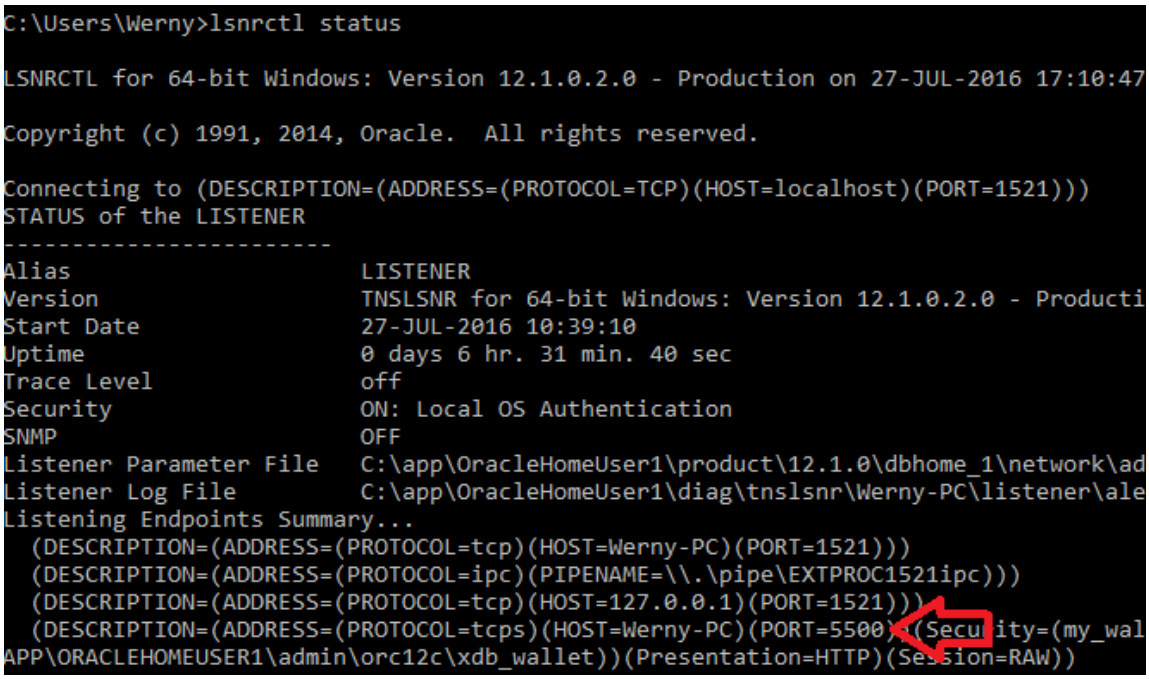
Nyní bude odeslán stejný SQL dotaz, ale tentokrát se o jeho vykonání bude starat komponenta IMCS. Jak je vidět na obrázku č. 38, vykonání tohoto dotazu s použitím sloupcového formátu trvalo jen 0.07 vteřin. Pro kontrolu bude odeslán dotaz ještě jednou. Jak je vidět, nic se nezměnilo a opět byla naměřena hodnota 0.07 vteřin. Lze tedy říci, že provedení SQL dotazu s pomocí IMCS je několikrát rychlejší než pomocí komponenty Buffer cache. Přesněji byl SQL dotaz pomocí komponenty IMCS vykonán 23-42x rychleji.

2.5 Další užitečné příklady

Poslední kapitola praktické části obsahuje několik užitečných příkladů, které se mohou hodit při správě Oracle Database 12c či obecně při správě databází. Bude zde například předvedeno, jak zjistit číslo portu, na kterém lze komunikovat s nástrojem EM Database Express, jakými způsoby lze restartovat databázovou instanci či jak povolit ukázkové SH schéma.

2.5.1 Jak zjistit číslo portu EM Database Express

Po instalaci softwaru Oracle Database 12c se v posledním kroku zobrazí url adresa včetně čísla portu, na kterém je nástroj EM Database Express k dispozici. Pokud však někdo upřednostňuje jiný nástroj pro správu a monitorování databáze, nemusí si číslo portu pamatovat. Prvním ze způsobu, jak zjistit číslo portu, je pomocí příkazové řádky systému Windows. Po zadání příkazu „lsnrctl status“ se zobrazí výpis z kontrolního posluchače Oraclu, kde kromě jiného nastavení lze nalézt také nastavení pro službu EM Database Express. Číslo portu a kde ho nalézt je vyznačeno na obrázku č. 39 pod tímto textem.



Obrázek 39: Ukázka nastavení posluchače Oracle, číslo portu EM Database Express

Druhým způsobem, jak lze zjistit číslo portu, je použitím nástroje SQL*Plus a příkazu select dbms_xdb_config.getHttpsPort() from dual. Výsledkem je opět stejné číslo portu jako v předchozím případě, tedy 5500.

2.5.2 Jak restartovat databázovou instanci

Databázová instance, která je nainstalovaná na notebooku či stolním počítači, může často zabírat více než polovinu fyzické paměti. Proto je dobré vědět, pokud zrovna s databází nepracuji, jak ji vypnout. Jindy, například z důvodu změny inicializačních parametrů, je zase třeba databázi restartovat. Po instalaci databázového systému do prostředí Windows 7 je databáze defaultně umístěna do tzv. programů po spuštění, což znamená, že se databázová instance spouští vždy při startu počítače. Proces, který zajišťuje běh této instance, se nazývá oracle.exe a lze ho najít například pomocí nástroje perfmon v prostředí operačního systému Windows. Pro ukončení databázové instance stačí tento proces ukončit. Vypnout a zapnout databázi lze pomocí nástroje Administration Assistant for Windows, který je součástí instalačního balíčku Oracle Database. V tomto graficky přívětivém prostředí stačí jednoduše kliknout pravým tlačítkem myši na databázovou instanci a zvolit, zda databázi vypnout či zapnout. Klasickým způsobem jak vypnout a opětovně zapnout databázovou instanci je pomocí jakéhokoliv nástroje, který podporuje dotazovací jazyk SQL. Zadáním příkazu SHUTDOWN se nejprve databáze uzavře, následně se odpojí a ukončí celá instance. Při startu pomocí příkazu STARTUP se databázová instance opět spustí. Po naalokování paměti se databáze připojí a konečně přejde do stavu OPEN. Restartování pomocí SQL dotazů znázorňuje následující obrázek č. 40.

```
SQL> shutdown;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
SQL>
SQL> startup;
ORACLE instance started.

Total System Global Area 4294967296 bytes
Fixed Size                  3053880 bytes
Variable Size             771753672 bytes
Database Buffers         2432696320 bytes
Redo Buffers               13721600 bytes
In-Memory Area           1073741824 bytes
Database mounted.
Database opened.
SQL>
```

Obrázek 40: Ukázka jak restartovat databázovou instanci pomocí nástroje SQL*Plus

2.5.3 Povolení ukázkového schématu SH

V ukázkovém příkladu pro práci s In-Memory bylo zapotřebí, vytvořit novou zásuvnou databázi, na které budou nainstalována ukázková schémata. Jak toho docílit bude předvedeno v tomto příkladu. Nová PDB bude vytvořena prostřednictvím nástroje Database Configuration Assistant. Po spuštění tohoto nástroje je třeba vybrat možnost Manage Pluggable Databases a následně vybrat kontejnerovou databázi, do které bude tato nová zásuvná databáze přidána. Uživatel zvolí databázi ORC12C. V dalším kroku musí uživatel vybrat způsob instalace nové databáze. Jelikož je potřeba vytvořit novou databázi a nainstalovat na ni ukázková schémata, uživatel musí vybrat poslední možnost, tedy Create Pluggable Database using PDB File Set. Následně je třeba vybrat soubory s metadaty, které obsahují ukázková schémata. Uživatel vybere soubor s názvem sampleschema.xml. Tím se vybere i druhý potřebný soubor. V dalším kroku uživatel zadá jméno databáze, například PDB_TEST. Je možné také vybrat administrátora pro tuto databázi, například SAM. V posledním kroku už je jen shrnutí nastavení. Po instalaci se bude databáze nacházet v otevřeném stavu a bude připravena pro čtení i zápis. To si lze snadno ověřit přihlášením se k databázi jako uživatel SYS pomocí SQL dotazu SELECT name, open_mode FROM v$pdbs. Nyní je ještě potřeba upravit soubor tnsnames.ora a restartovat databázovou instanci. Dalším krokem je povolení uživatelského účtu, který je rovněž součástí ukázkových schémat a kterým bude později uživatel k databázi PDB_TEST přistupovat. Uživatel se tedy přihlásí do databáze PDB_TEST jako společný uživatel SYS, aby odemkl uživatelský účet SH. Odemknutí lze provést příkazem ALTER USER SH IDENTIFIED BY SH ACCOUNT UNLOCK. Tímto je uživatelský účet odemknut a uživatel se může pokusit připojit k databázi PDB_TEST prostřednictvím uživatele SH. Připojení lze provést příkazem CONN SH/SH@PDB_TEST.

```
SQL> conn sys/Oracle12c@pdb_test as sysdba;
Connected.
SQL> alter user sh identified by sh account unlock;

User altered.

SQL> conn sh/sh@pdb_test;
Connected.
SQL>
```

Obrázek 41: Ukázka odemknutí uživatelského účtu SH

Pro přístup k materializovaným pohledům je třeba uživateli SH přidělit patřičná oprávnění. Bude mu tedy přidělena role DBA, která obsahuje většinu administrativních oprávnění. Provést to lze prostřednictvím uživatele SYS a příkazu GRANT DBA TO SH.

Srovnání s Oracle Database 11g

Pokud bych se rozhodl, že si pořídím novější model mobilního telefonu, mohu s určitostí říci, že by byl lepší než ten stávající. Stejně tak tomu je i u nové verze databáze Oracle Database 12c ve srovnání s verzí předcházející, tedy Oracle Database 11g. Přestože se tato verze stále velmi hojně používá, mnoho zákazníků volí novější verzi 12c. Je to hlavně díky nové multitenantní architektuře, která kromě jiného také umožňuje spravovat databázi pomocí cloudu. Jak již bylo několikrát zmíněno, hlavní předností nové verze Oracle Database je nová multitenantní architektura, která se skládá z kontejnerové databáze, do které lze připojit přes 250 zásuvných databází. Mohu zde vyjmenovat několik výhod, které s použitím multitenantní architektury souvisí.

- Záplaty a vylepšení
  Aktualizace a záplaty jsou aplikovány přímo na hlavní kontejner, není nutné aktualizovat či záplatovat každou zásuvnou databázi zvlášť.
- Zálohování dat
  Zálohování dat je velmi důležitým bezpečnostním prvkem. Díky multitenantní architektuře již není nutné zálohovat každou databázi zvlášť, ale je možné zálohovat celý kontejner CDB jako jednu multitenantní databázi.
- Rychlé opravy a klonování
  Rychlé opravy a klonování, ať už pro účely testování, vývoje či diagnostiky problémů jsou vždy výzvou pro mnoho IT organizací. Správci databází věnují nejvíce času právě vytváření nových databází, klonování databází a přesunu databází mezi různými servery. Starší verze Oracle Database vyžadovaly k naklonování databáze mnoho kroků, které byly časově velmi náročné. Nyní však stačí jediný krok k tomu, aby byla požadovaná databáze úspěšně naklonována. Celé klonování pak proběhne téměř okamžitě.

Navíc hlavně díky technologii In-Memory, kterou tato nová verze Oracle Database disponuje, lze efektivně pracovat a zpracovávat tzv. Big Data.

Shrnutí výsledků

Nová verze databázové platformy Oracle Database 12c přináší mnoho nových technologií. Asi nejpřevratnější novinkou této verze je nová multitenantní architektura, díky které lze plynule přejit ke cloudovému řešení databází. Cloudu a cloudovým řešením databází se tato práce nevěnuje, ačkoliv je to hlavním symbolem této majoritní verze. Poslední vydání Oracle Database 12c (12.1.0.2) navíc ještě přichází s novou technologií In-Memory, která zásadním způsobem mění pohled na uchovávání dat v paměti.

Oracle Database 12c je postavena na nové multitenantní architektuře. Klíčové slovo multitenantní (ang. Multitenancy), lze volně přeložit jako „více-nájemná“. Jde o to, že na jednom fyzickém serveru je spuštěna pouze jedna aplikace, která je sdílena mezi více klientů. Tato databáze se také někdy označuje jako multitenantní kontejnerová databáze. Pod kontejnerovou databází si lze představit jakousi hlavní databázi, která poskytuje prostředky tzv. zásuvným databázím. Zásuvné databáze se po připojení ke kontejnerové databázi o tyto prostředky musí dělit. Kontejnerová databáze může obsahovat jednu, mnoho, ale nemusí obsahovat žádnou zásuvnou databázi. Díky tomu, že mnoho zásuvných databází sdílí prostředky kontejnerové databáze, dochází k tzv. konsolidaci databáze. Konsolidace představuje import mnoha zásuvných databází do jedné kontejnerové databáze, čímž se výrazně šetří technické vybavení a tím pádem i finanční prostředky na provoz mnoha databázových systémů. Díky multitenantní architektuře lze také spravovat mnoho zásuvných databází najednou, což urychluje práci a šetří čas, který by jinak musel být vynaložen na správu těchto databází.

Na začátku teoretické části byla stručně popsána nová multitenantní architektura. Byl zde například vysvětlen rozdíl mezi kontejnerovou databází a zásuvnou databází. Dále byla popsána logická a fyzická struktura kontejnerové databáze a vysvětlen princip rozdělení datového slovníku. V další kapitole teoretické části byla probrána architektura paměti Oracle Database 12c, kde byly popsány dvě základní oblasti paměti, globální systémová a globální programová oblast. Poslední kapitola teoretické části se věnovala nové technologii In-Memory. Zde byly popsány souvislosti s In-Memory a sloupcovým formátem uchovávání dat v paměti.

Praktická část práce se věnovala vybraným funkcionalitám, které byly popisovány v části teoretické. Nejprve byly představeny užitečné nástroje, pomocí kterých lze spravovat či monitorovat databázový systém Oracle. Následně byla provedena ukázková instalace softwaru Oracle Database 12c, v rámci které byla vytvořena kontejnerová databáze ORC12C. Další kapitola byla věnována práci se zásuvnými databázemi. Pomocí nástroje SQL Developer tak bylo předvedeno, jak je nyní správa zásuvných databází snadná. Následující část se věnovala technologii In-Memory, například nastavení inicializačních parametrů či připravení databázového objektu pro zpracování komponentou SGA paměti zvanou In-Memory Column Store. Na závěr zde bylo předvedeno několik užitečných příkladů.

Závěry a doporučení

Nová verze databázové platformy Oracle Database 12c přináší spoustu nových technologií a přístupů, kterým se lze věnovat. Tato práce se pochopitelně nemohla věnovat všem těmto technologiím, ale věnovala se pouze těm, které byly autorem práce shledány za důležité.

Tato bakalářská práce popisuje některé vybrané technologie, které sebou přináší nová verze Oracle Database 12c. Mohla by být proto přínosem pro studenty, kteří se o tyto technologie zajímají. Dále by se mohla hodit začínajícím administrátorům databázových systémů Oracle, kteří by se mohli inspirovat z některých praktických příkladů či ukázek.

V budoucnu se lze zabývat například již zmiňovaným cloudovým řešením databází, či dalšími funkcionalitami, které se do této práce nevešly.

Seznam použité literatury

[1] KYTE, Tom a Lance ASHDOWN. *Oracle Database Online Documentation 12c Release 1 (12.1): Introduction to the Multitenant Architecture* [online]. ORACLE. 2014 [cit. 2016-02-15]. Dostupné z: http://docs.oracle.com/database/121/CNCPT/cdbovrvw.htm#

[2] *Creating Oracle 12c Multitenant Container Database: Multitenant Architecture* [online]. 2015 [cit. 2016-02-15]. Dostupné z: https://avdeo.com/2015/01/16/creating-oracle-12c-multitenant-container-database/

[3] KYTE, Tom a Lance ASHDOWN. *Oracle Database Online Documentation 12c Release 1 (12.1): Oracle Database Structure and Storage* [online]. ORACLE. 2014 [cit. 2016-02-15]. Dostupné z: http://docs.oracle.com/database/121/ADMIN/part2.htm

[4] FOGEL, Steve. *Oracle Database Online Documentation 11g Release 2 (11.2): Logical Storage Structures* [online]. ORACLE. 2013 [cit. 2016-02-15]. Dostupné z: http://docs.oracle.com/cd/E11882_01/server.112/e40540/logical.htm#CNCPT004

[5] PAVLOVIĆ, Zoran a Maja VESELICA. *Architecture Changes and New Security Features in Oracle Database 12c*[online]. 2013 [cit. 2016-02-15]. Dostupné z: http://www.slideshare.net/ZPavlovic/database-12c-23625983

[6] KYTE, Tom a Lance ASHDOWN. *Oracle Database Online Documentation 12c Release 1 (12.1): Data Dictionary and Dynamic Performance Views* [online]. ORACLE. 2014 [cit. 2016-02-15]. Dostupné z: https://docs.oracle.com/database/121/CNCPT/datadict.htm#CNCPT002

[7] KYTE, Tom a Lance ASHDOWN. *Oracle Database Online Documentation 12c Release 1 (12.1): Memory Architecture*[online]. ORACLE. 2014 [cit. 2016-02-15]. Dostupné z: http://docs.oracle.com/database/121/CNCPT/memory.htm#CNCPT1234

[8] KYTE, Tom a Lance ASHDOWN. *Oracle Database Online Documentation 12c Release 1 (12.1): Managing Memory*[online]. ORACLE. 2014 [cit. 2016-02-15]. Dostupné z: http://docs.oracle.com/database/121/ADMIN/memory.htm#ADMIN11011

[9] DYKE, Julian. *SGA Internals: Granules* [online]. 2005 [cit. 2016-02-15]. Dostupné z: http://www.slideshare.net/sergkosko/sga-internals

[10] COLGAN, Maria. *Oracle Database In-Memory* [online]. 2015 [cit. 2016-02-15]. Dostupné z: http://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-in-memory-2245633.html

[11] *Oracle In-Memory Database Option* [online]. 2014 [cit. 2016-02-15]. Dostupné z: https://www.linkedin.com/pulse/20141128065216-11765692-oracle-in-memory-database-option-the-game-changer

[12] KYTE, Tom a Lance ASHDOWN. *Managing Memory: Using the In-Memory Column Store* [online]. ORACLE, 2014 [cit. 2016-08-01]. Dostupné z: https://docs.oracle.com/database/121/ADMIN/memory.htm#ADMIN14239

[13] *Oracle Database In-Memory: Dual Format Database* [online]. ORACLE, 2014 [cit. 2016-08-01]. Dostupné z: http://www.oracle.com/technetwork/database/in-memory/overview/index.html

[14] *Oracle Database In-Memory: SIMD VECTOR PROCESSING* [online]. OTECH MAGAZINE, 2014 [cit. 2016-08-01]. Dostupné z: http://otechmag.com/magazine/2015/summer/mahir-m.-quluzade.html

[15] *A History Of Oracle* [online]. DSP - Database Services Partner, 2014 [cit. 2016-08-11]. Dostupné z: http://www.slideshare.net/dsp_uk/history-of-oracle-31435392

[16] ŠEDA, Miloš. *Databázové systémy* [online]. Brno, 2002, 4 [cit. 2016-08-11]. Dostupné z: http://www.uai.fme.vutbr.cz/~mseda/DBS02_BS.pdf

Seznam obrázků

Seznam tabulek

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2015/2016

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Werner Jan	5.května 112, Meziměstí	I1200964

TÉMA ČESKY:

Nové technologie Oracle Database 12c

TÉMA ANGLICKY:

New technologies in Oracle Database 12c

VEDOUCÍ PRÁCE:

Ing. Barbora Tesařová, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cílem práce je představit nové technologie v Oracle Database 12c a to konkrétně novou architekturu Oracle Multitenant. Dále pak popsat a uvést na příkladech práci s kontejnerovými a zásuvnými databázemi včetně jejich předností a úskalí.

Na tomto základě porovnat s předchozí verzí Oracle Database 11g

Osnova:
1. Úvod
2. Cíl práce
3. Metodika
4. Multitenantní architektura
5. Kontejnerové a zásuvné databáze
6. Připojení do Cloudu
7. Práce se zásuvnými databázemi
8. Závěr

SEZNAM DOPORUČENÉ LITERATURY:

ORACLE. 2014. Database Administrator's Guide [online]. [cit. 2015-10-23]. Dostupné z: https://docs.oracle.com/database/121/ADMIN/toc.htm

HALL, Tim. 2014. Oracle 12c Articles [online]. [cit. 2015-10-23]. Dostupné z: https://oracle-base.com/articles/12c/articles-12c

Podpis studenta: ..... **Datum:** 15.10.2015

Podpis vedoucího práce: ..... **Datum:** 15.10.2015