Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Návrh a realizace Oracle Standby technologie v prostředí ZČU

Plzeň, 2018 Bc. Michal Čermák

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 17.května 2018

Bc. Michal Šermák

Poděkování

Rád bych touto cestou poděkoval vedoucímu práce panu Ing. Petrovi Jiroušekovi za odborné vedení mé diplomové práce, jeho čas a cenné informace a za jeho ochotu při vedení práce. Můj vděk patří také panu Ing. Martinovi Zímovi, PhD., konzultantovi práce, za udělování cenných rad a připomínek, které mi pomohly při zpracování práce a za jeho ochotu věnovat se častým schůzkám.

Abstract

Name: Design and implementation of Oracle Standby technology in environment UWB

Standby technology is part of product Oracle Data Guard developed by Oracle Corporation. This product provides data availability and disaster recovery.

This thesis aims to present ability of Oracle Data Guard. Theoretical part of the paper describes theory of data protection and description of Oracle Data Guard with selection of suitable implementation for data protection of university information system. The practical part is based on the theoretical part and describes design and implementation solution of product Oracle Data Guard inclusive of tests of functions. In conclusion, what product offers common user and the result of thesis are evaluated.

Abstrakt

Název: Návrh a realizace Oracle Standby technologie v prostředí ZČU

Standby technologie je součástí produktu Oracle Data Guard od společnosti Oracle. Tento produkt je zaměřen na zajištění dostupnosti a ochrany dat.

Předložená práce prezentuje možnosti Oracle Data Guard. V úvodní části je charakteristika problematiky ochrany dat a popis Oracle Data Guard s výběrem vhodné implementace, která bude nejlépe chránit data univerzitního informačního systému. Na tomto teoretickém základě staví praktická část, popisující návrh a implementaci konkrétního řešení produktu Oracle Data Guard včetně následných testů funkčnosti implementovaného řešení. V neposlední řadě je uvedeno, co tento produkt nabízí z pohledu běžného uživatele. Na závěr je uvedeno zhodnocení implementovaného řešení.

Obsah

1 Úvod

Ochrana dat představuje významnou část IT, kterou musí řešit všechny typy organizací. Ať už se jedná o malé nebo velké podniky, školy, státy, banky atd., ti všichni potřebují ke svému fungování data, která je nutno stále řešit. Kriticky důležitá je nejenom ochrana dat ale i zajištění neustálého přístupu k datům. Nedostupnost nebo v horším případě ztráta dat může mít nedozírné následky. Z pohledu obchodních společností může dojít ke ztrátě zisku a reputace, z pohledu univerzit může dojít k přerušení jejího běžného fungování.

Západočeská univerzita v Plzni využívá pro svá data databázi od společnosti Oracle. Oracle nabízí pro zajištění ochrany a dostupnosti dat několik produktů. Tato práce se věnuje jednomu konkrétnímu produktu a to Oracle Data Guard. Tento produkt zajišťuje dostupnost i ochranu dat prostřednictvím nasazení záložních (standby) databází.

Cílem této práce je vybrat v rámci produktu Oracle Data Guard vhodný typ záložní databáze pro prostředí Západočeské univerzity, poté provést implementaci technologie Oracle Data Guard s vybraným typem záložní databáze a otestovat funkčnost implementované technologie.

Úvod textu je zaměřen na popis ochrany databází a na produkty společnosti Oracle, IBM a Microsoft řešící tuto problematiku.

Po uvedení do problematiky následuje seznámení s produktem Oracle Data Guard. V této části je popis funkcí Oracle Data Guard, posléze jsou rozebrány typy záložních databází s výčtem výhod a nevýhod jednotlivých typů. Následně jsou srovnány potřeby univerzitního informačního systému s možnostmi jednotlivých záložních databází.

Ve zbývajících kapitolách se nalézá popis implementace vybrané záložní databáze a průběhu provádějících testů. Tyto testy se zaměřují především na funkčnost a rychlost tohoto Oracle řešení. V neposlední řadě je popsáno, jaké výhody tento produkt nabízí z pohledu běžného uživatele.

2 Databázové systémy a ochrana dat

2.1 Databázový systém

V souasném informaním vku existuje záznam v elektronické formtém o každém lovku, nap. jeho jméno, datum narození v lékaských záznamech, záznam provedeného nákupu v eshopu, íslo bankovního útu v systému zamstnavatele, atd. Pro uložení tchto informací slouží databázový systém. Tento systém se skládá z databáze a z database management systemu (DBMS) [27].

Databáze je kolekce logicky souvisejících dat a popis tchto dat [32]. Popisem dat je myšleno, že databáze obsahuje meta-data, která popisují data uložená v databázi. DBMS je sada program slouící k získávání, ukládání a modifikování dat. Prostednictvím aplikací pevádí data z fyzické reprezentace do logické a zpt. DBMS je dleitý vždy, když je poteba ukládat a pistupovat k velkému objemu dat. DBMS je základním kamenem informaních systém. I když webové a aplikaní servery nebo informaní systémy jsou dleitou souástí získávání a zpracování dat, tak je to DBMS, které spravuje úložišt obsluhující velká množství dat. Navíc když nap. množství dat sbíraných ze sociálních sítí exponenciáln roste. Vtšina komerních DBMS krom zmínných základních funkcí má i další schopnosti jako je zajištní pístupu k datm více uživatelm souasn, ízení pístupových práv, kontrolování platnosti dat nebo obnovu pi selhání systému bez ztráty dat [27].

Všechny verze Oracle databází využívají relaní DBMS (RDBMS) model pro ukládání dat do databáze. Pro pístup k databázím je využíván jazyk SQL. SQL umí pracovat s relaními databázemi. Pi využívání SQL není poteba psát komplexní poítaové programy nebo znát fyzické umístní hledaných dat. Relaní model umožuje jasn identifikovat data a vztahy mezi tmito daty. Pi vytváení relaní databáze je tedy poteba urit, co data znamenají, a jaké jsou mezi nimi vztahy.

V relaní databázi jsou data organizována do entit. Entity jsou ásti informace. Entita pedstavuje prvek reálného svta a je popsána svými vlastnostmi (atributy). Atributy mohou být nap. email, adresa, plat, atd. [4]. V relaních databázích se obvykle pro entity a atributy užívá oznaení tabulky a sloupce [28].

Nejvtší pednost relaních databází je vytváení vztah mezi tabulkami. V pípad dvou a více tabulek (entit) je možné mezi nimi vytvoit vztah. V pípad, že se v relaní databázi nacházejí dv tabulky (entity) nap. zamstnanec, která obsahuje

seznam osob pracujících v podniku, a email, která obsahuje seznam emailů, lze mezi nimi vytvořit vztah typu 1:N, umožňující přiřazení více emailových adres k jednomu konkrétnímu zaměstnanci. Aby bylo možné takovéto vztahy vytvořit, je nutné vytvořit pro tyto tabulky primární a cizí klíče. Primární klíč je atribut nebo množina atributů (složený klíč), jenž umožňuje určit unikátní řádek s daty v tabulce (v příkladu se zaměstnanci a emaily se primární klíč nachází v tabulce zaměstnanec, a umožňuje tak určit konkrétního osobu). Cizí klíč entity je primární klíč jiné entity (v příkladu se nachází se v tabulce email). Tyto dva klíče označují stejný atribut a spolu vytvářejí spojení mezi entitami (tabulkami). [4, 28]

Každý RDBMS podporuje různé typy datových objektů. Oracle 12c podporuje celou řadu objektů jako jsou tabulky, pohledy, balíky (packages), triggery, atd. [4].

2.2 Ochrana databází

Téměř každý aplikační software potřebuje ke své práci data, která jsou obvykle uložena v nějaké databázi. K této databázi je pak nutné zajistit dostupnost, v co nejvyšší možné míře a neustále zálohovat její data pro případ nějaké nenadálé události, při níž by došlo ke ztrátě dat. Zajistit vysokou dostupnost dat a jejich ochranu nepotřebují jenom společnosti, jejichž hlavním cílem je zisk, ale i všechny ostatní organizace, jejichž cílem není tvořit profit, jako jsou např. univerzity. V případě nedostupnosti dat a tedy i aplikací, které s těmito daty pracují, může to mít závažné následky. Z pohledu firmy může dojít ke ztrátě reputace, odchodu zákazníků ke konkurenci, a tím ve výsledku ke ztrátě zisku. I univerzity potřebují zajistit pro svůj provoz dostupnost svých služeb a chránit svá data a data uživatelů. Dostupná řešení starající se o tuto problematiku spadají do kategorie produktů zajišťujících vysokou dostupnost a disaster recovery databáze. (produkty spadající do kategorie disaster recovery zajišťují obnovu IT systému po živelných pohromách a katastrofách, blíže vysvětleno v podkapitole 2.2.2) [1, 33].

Diplomová práce se věnuje především robustnímu řešení Data Guard od společnosti Oracle, jenž zajišťuje vysokou dostupnost i disaster recovery. Na trhu ale existuje vícero produktů týkajících se ochrany a dostupnosti databáze. Každá společnost nabízí svá specifická řešení, která lze většinou využít naplno pouze s jejich databázovými systémy. Tito výrobci v rámci konkurenčních bojů neustále inovují své

produkty a snaží se přidávat do svých řešení vlastnosti, jež by je odlišily a přinesly konkurenční výhodu.

V této práci jsou pro srovnání uvedeny některé další robustní produkty konkurenčních společností. Následující odstavce se věnují stručnému přehledu dalších možností řešení vysoké dostupnosti a disaster recovery.

2.2.1 Vysoká dostupnost

Jednoduše řečeno, vysoká dostupnost (High availability) znamená, že systém bude zapnut a dostupný, kdykoliv se uživatel k němu připojí. V dnešním světě je potřeba, aby počítačové systémy fungovaly nepřetržitě. Bohužel, ale žádný IT systém není dokonalý a trpí problémem výpadků. Administrátoři se musí potýkat s problémy způsobenými softwarovými a hardwarovými selháními, chybami zaviněnými lidmi a přírodními katastrofami. Tyto příčiny mohou zapříčinit výpadek, během něhož systém neposkytuje uživatelům své služby nebo je poskytuje jen v omezené míře. Předcházejícím pokrytím a vykompenzováním těchto výpadků se zabývají produkty pro vysokou dostupnost. U společnosti Oracle se lze setkat s pojmem Maximum Availability Architecture (MAA), představující nejlepší návrhy a postupy (best practices) pro dosažení vysoké dostupnosti [1, 33].

Výpadky lze rozdělit na plánované a neplánované. Plánované výpadky jsou předem připravovány. Tyto výpadky jsou vyvolány uměle a mají stanovený časový plán jejich průběhu. Obvykle se provádějí z důvodů údržby nebo upgradu softwarových a hardwarových součástí [1, 33].

Neplánovaný výpadek je nepředvídatelná událost. Může vzniknout z různých příčin od softwarové chyby až po požár v budově. Nejlepší obranou proti neplánovaným výpadkům je snažit se jim předcházet důsledným monitoringem všech součástí systému a vnějších vlivů, které na systém mohou působit. Pro případ výpadku je dále potřeba mít předem připravené postupy, jak situaci řešit. Výpadkům se také nechá zabránit i správným navržením infrastruktury. Je potřeba zajistit, aby žádná z komponent infrastruktury nebyla tzv. Single Point of Failure. Taková komponenta nemá redundantního náhradníka a v případě jejího výpadku selže celý systém [33].

2.2.2 Disaster recovery

Druhý okruh pokrývá oblast zvanou disaster recovery. Obsahy pojmů disaster recovery a vysoké dostupnosti se do určité míry shodují. Obě možnosti mají společné, že zajišťují nepřetržitou dostupnost, přístup k datům a jejich integritu, i když z různých důvodů. Disaster recovery představuje proces obnovy databáze v případě jejího nepředvídatelného selhání jako jsou přírodní pohromy, ale i člověkem zaviněné katastrofy. Obvykle se zajišťuje obnova celého systému v jedné lokalitě. Pro zajištění disaster recovery je tedy nutné vybudovat podpůrný systém na jiném místě [1, 33].

2.3 Oracle

Společnost Oracle je jedna z hlavních společností vyvíjejících databázové systémy. Působí ve 175 zemích světa a má téměř půl milionu zákazníků. Mezi nejvýznamnější zákazníky patří např. velké bankovní instituce jako Merril Lynch a Citigroup, eBay a německá letecká společnost Lufthansa [35]. Oracle nabízí v oblasti vysoké dostupnosti a disaster recovery několik produktů, které je možné kombinovat. Zejména se jedná o technologie Real Application Cluster (RAC), Oracle Recovery Manager (RMAN), Oracle Flashback Technologie a Oracle Data Guard, jemuž se věnuje tato práce a je blíže rozebrán v následující kapitole [1].

RAC je clusterová technologie nabízející řadu možností pro kritické databázové systémy. Navazuje na předchozí technologii Parallel Server. Jedná se o sloučení dvou a více databázových instancí, s možností nasazení na více serverů sdílejících jedno datové úložiště. Tato skupina pak zvenčí vypadá jako jeden celek. Tyto instance umožňují rozdělit zátěž z příchozích uživatelských požadavků mezi sebe. V případě selhání jedné z databázových instancí je zátěž odkloněna na ostatní instance. RAC není zcela závislá na technologiích Oracle, ale lze ji nasadit na uzly clusteru, které využívají produkty jiných dodavatelů. Pro plné využití funkčnosti RAC je však software Oracle clusteware nutný [1, 34].

Pro zajištění ochrany dat nabízí Oracle funkcionalitu Recovery Manager (RMAN). Jedná se o softwarový nástroj s příkazovou řádkou. Zajišťuje zálohování s minimálním zatížením produkční databáze a rychlou obnovu databáze v případě ztracených dat. RMAN může být použit pro usnadnění administrace Oracle Data Guard konfigurace. Především urychluje vytvoření záložních databází [1, 34].

Pro zrcadlení souborů poskytuje Oracle databáze funkci Automatic Storage Manager (ASM). ASM je nástroj pro správu diskového prostoru, který ochraňuje data, v případě selhání databázového úložiště. Pro svoji funkci využívá diskových skupin (disk failure groups). Tyto skupiny obsahují jednotlivé disky, ze kterých je vybudován souborový systém. Data jsou pak rozdělena na menší dílky a uložena na vícero disků. ASM zvyšuje bezpečnost a rychlost práce s daty [1, 34].

Další technologie je Flashback databáze, která dokáže vrátit databázi do stavu, kdy byla v pořádku. Použít tuto funkci s Data Guardem je velmi žádoucí, protože pomáhá chránit databázi před lidskou chybou. Kombinace těchto technologií dokáže automaticky opravovat vzniklé problémy [1, 34].

2.4 Microsoft

Dalším výrobcem databázových systémů je společnost Microsoft, která je významný poskytovatel operačních systémů pro osobní počítače a servery. Již dlouhou dobu poskytuje i vlastní databázový systém. Její nejnovější verze je Microsoft SQL Server 2017. Ke svému databázovému systému nabízí řadu produktů pro vysokou dostupnost a disaster recovery [38].

K některým serverovým systémům Microsoft dodává vlastní clusterware software (Microsoft clustware server). Konkrétní část zajišťující vysokou dostupnost se nazývá Failover Clustering (WSFC). Obdobně jako u clusterových řešení ostatních výrobců, se jedná o skupinu několika serverů, které vypadají navenek jako jeden. Aplikace běží pouze na jednom serveru. V případě jeho výpadku jsou služby spuštěny na jiném serveru v clusteru. Všechny aplikace fungující v clusteru mají definovaný jeden primární a alespoň jeden sekundární server, na kterém začnou fungovat v případě výpadku primárního serveru [15, 20, 33].

Další možností, která zajišťuje vysokou dostupnost a nabízí řešení pro disaster recovery, je funkcionalita AlwaysOn. Toto řešení je dostupné od verze 2012 a pouze v Enterprise edici (EE). AlwaysOn si nebere za úkol sama poskytnou vysokou dostupnost a řešení pro disaster recovery, ale umožňuje vylepšit a zefektivnit již stávající řešení společnosti Microsoft. Kombinace AlwayOn nasazená s Windows Server Failover Clustering se nazývá AlwaysOn Failover Cluster Instance (FCI). Další funkcionalita, se kterou je AlwaysOn integrováno je Availability Groups, kde umožňuje jednodušší konfiguraci databázových skupin [15, 20, 33].

Availability Groups technologie zajiš uje udržování záložní databáze plnì synchronní s produkční databází. Tyto repliky produkční databáze mohou být až tyři. Zmìny provedené v produkční databázi jsou posílány na záložní server, kde jsou aplikovány na databázi. Data je možné pøenášet synchronnì i asynchronnì. Volba režimu pøenosu urèuje rychlost a úroveò ochrany celého øešení. V pøípadì selhání primární databáze lze zmìnit jednu z replik na primární databázi [15, 20, 33].

Další podobné øešení využívající záložní databáze je Database Mirroring. Spoleènost Microsoft na svých stránkách doporuèuje využívat øešení Availability Group, nebo dochází k útlumu vývoje produktu Database Mirroring a obì øešení jsou si velmi podobné. V tomto øešení jsou dva servery spojeny do tzv. session (database mirroring session). V této session vystupuje server s produkční databází jako primární, jenž se nazývá principal server a druhý server se záložní databází, který je nazýván mirror server. Tato zrcadlená databáze mùže být umístìna na stejném serveru nebo na jiném serveru v jiném datovém centru. V pøípadì selhání produkční databáze dojde k pøepnutí rolí a ze záložní databáze se stane nová produkční databáze. Database Mirroring stejnì jako Availability Group nabízí tøi typy failoveru a to manuální, automatický a vynucený. Pro automatický failover je nutné mít v provozu další server, který monitoruje dostupnost primární databáze (witness server) a v pøípadì nedostupnosti automaticky spustí failover [15, 20, 33].

Poslední zmínìná technologie firmy Microsoft je Log Shipping. Je to jednoduchá funkcionalita pro zálohování logù produkční databáze. V mnoha smìrech se podobá technologii Database Mirroring. Log Shipping zajiš uje pravidelné zálohování transakèních logù. Nepøetržitì distribuuje transakèní logy na záložní servery, na kterých jsou tyto zálohy obnoveny. Navíc mùže být nasazen další server, který celý proces sleduje a sbírá informace o zálohování a obnovì [15, 20].

2.5 IBM

Spoleènost International Business Machines Corporation (IBM) je další významnou spoleèností na trhu databázových systémù. IBM poskytuje databázový systém IBM DB2 a pro jeho zabezpeèení nabízí technologii High Availability Disaster Recovery (HADR). Jak název napovídá, toto øešení pokrývá dostupnost i pøípady selhání databáze. Nabízí tedy jak vysokou dostupnost v rámci jedné lokality, tak i možnost disaster recovery mezi vzdálenými lokalitami [33].

HADR nepřetržitě v reálném čase replikuje data na ostatní záložní databáze. V případě selhání nebo nedostupnosti produkční databáze, jedna ze záložních databází převezme funkci produkční databáze. HADR umožňuje mít v nastavení na jednu primární databázi až tři hlavní standby databáze (principal standby database). IBM umožňuje nasadit více záložních databází. Ty jsou brány jako pomocné standby databáze (auxiliary HADR standby database), nemající plnou funkčnost hlavních standby databází. Pomocné standby databáze neumožňují automatickou změnu databáze ze standby na primární [33].

HADR podporuje synchronní, asynchronní, tzv. NEARSYNC a SUPERSYNC přenos transakčních logů k aktualizaci záložních databází. Přenosy se rozlišují podle požadavků, na které čekají než potvrdí probíhající transakci. Synchronní vyžaduje potvrzení o uložení logů na disku standby a asynchronní čeká pouze na zprávu o jejich odeslání. NEARSYNC přenos čeká na replikaci logů do paměti standby a při přenosu SUPERSYNC není před potvrzením transakce provádějící na produkční databázi vyžadováno žádné potvrzení. Pomocné standby běží vždy v režimu přenosu SUPERSYNC [33].

Technologii HADR je možné nasadit s dalším řešením zvaným PureScale. Jedná se o clusterovou technologii. IBM nabízí stejně jako Oracle vlastni clusteware software PowerHA. Toto řešení lze však nasadit pouze s operačním systémem AIX. PowerHA zajišťuje distribuování požadavků na databázi mezi jednotlivé servery, zjišťuje dostupnost jednotlivých uzlů clusteru, atd. V případě výpadku uzlu přesměruje požadavky na ostatní servery [33].

3 Oracle Data Guard

3.1 Základní popis

Diplomová práce se vnuje pedevgím produktu firmy Oracle, protoge databázový systém této spolenosti je nasazen v prostedí Západoeské Univerzity. Data Guard je soustí produktu spolenosti Oracle od verze 7.3. V této verzi byl znám pod názvem standby databáze. Ag od verze 9i ji lze nalézt pod pojmem Oracle Data Guard (ODG). Základní funkce Data Guardu bhem let zstala stejná, tj. udrgování zálogní (standby) databáze, která je synchronizovaná s produkní databází. Z celkového pohledu celé softwarové ešení Data Guard doznalo od verze 7.3 k soustasnosti adu zlepgení [2, 3, 30]. Ve verzi 12.1.0.2.0 je ODG (stejnj jako RMAN a RAC) soustí databázového softwaru pouze v edici Enterprise. Dalgími nabízenými databázovými edicemi Oracle jsou Standart nebo Express, ale jejich mognosti nasazení jsou omezené a vtšinu ešení pro disaster recovery a vysokou dostupnost neobsahují [31].

Jedná se o ešení pokrývající oblast ochrany a obnovy dat i celé databáze a zajitní vysoké dostupnosti databáze a jejích dat. Data Guard umoguje adit, monitorovat, spravovat tzv. Data Guard konfiguraci. Tato konfigurace obsahuje dv hlavní komponenty. Jedná se o primární databázi, která pedstavuje databázi, jíg je poteba chránit, obvykle se jedná o produkní databázi a ze standby databáze, která slougí jako zálogní databáze. Data Guard umoguje mít ve své konfiguraci ag 30 standby databází rzných typ (fyzická, logická, snapshot), jeng vgechny obsahují data primární databáze. Tyto databáze obvykle bgí v rzných lokalitách a je mogné je provozovat na rzném hardwaru. Rozdlení standby databází do rzných geografických míst sniguje riziko v pípad, ge vznikne njaká pohroma v místprimární databáze [1, 2, 3].

Prvotním úelem ODG je udrgování synchronizace standby databází s primární databází. Kdyg nkdy nastane situace, ge primární databáze selge, Data Guard automaticky pepne jednu ze standby databází na primární databázi, tj. jedna ze standby databází pevezme vgechny úkony, které vykonávala primární databáze, anig by doglo ke ztrátdat. V pípadvýpadku primární databáze zpsobeného nap. porugením dat, Data Guard dokáge pomocí dat ze standby databáze automaticky opravit primární databázi. Výpadek zpsobený pepnutím na novou primární databázi trvá v adu sekund

aŕ minut. Uŕivatelé pak pŘistupují k této nové produkční databázi naprosto stejnŏ jako v pŘípadŏ pŘedchozí primární databáze [1, 2, 3].

V pŘípadŏ, ŕe databáze uvnitŘ ODG konfigurace běží bez obtíží, Data Guard umoŕňuje ke zvýšení dostupnosti dat část zátěŕe tvoŘené na primární databázi pŘerozdělit na standby databáze. Další moŕnosti vyuŕití standby databáze jsou pro reportování nebo testování databáze [2, 3].

3.2 Architektura Oracle databáze

Tato kapitola se věnuje pŘedevším části architektury řídící ukládání dat. Zprvu je potŘeba oddělit pojmy, které se pojí s Oracle architekturou, a to databázi a databázovou instanci. Oracle instance se skládá z mnoŕiny Oracle procesů, které běží na pozadí operačního systému a skupin vyrovnávacích pamětí. Kaŕdá databáze by měla mít minimálně jednu instanci, která je k ní pŘipojena. Spolu pak obě Oracle části běží na jednom serveru. V pŘípadě moŕnosti nasadit více instancí na jednu databázi nabízí Oracle technologii RAC, která umoŕňuje logicky i fyzicky oddělit instanci od databáze. Technologie RAC je blíŕe popsána ve druhé kapitole. Databáze je skupina souborů, v níŕ jsou uloŕena data. Instance i databáze vyuŕívají tři základní hardwarové součásti počítače. Instance vyuŕívá procesor a operační paměť a databáze vyuŕívá pevný disk [4, 27].

Celý proces uloŕení dat fyzicky na disk začíná potvrzením commit nějaké změny, ať uŕ se jedná `INSERT`, `UPDATE`, `DELETE` či vytvoŘení celé tabulky. Ve verzi Oracle 12c (Oracle 12.1.0.2.0) se o zápis dat do datových souborů databáze stará proces Database Writter (DBW, dŘíve DBWR), který je součástí databázové instance. Informace, co všechno má do datových souborů zapsat, získává z vyrovnávací paměti. Kdyŕ uŕivatel potvrdí změnu dat pŘíkazem `UPDATE`, DBW najde datovou jednotku zvanou databázový blok (datový blok je nejmenŕí jednotkou dat v Oracle databázi) na fyzickém disku, ve kterém se nalézají poŕadovaná data, a umístí ji do speciální vyrovnávací paměti (buffer) v operační paměti serveru. V tomto bufferu pak dochází ke změně dat. Posléze jsou pozměněná data procesem DBW zapsána na disk. Oracle verze 12.1.0.2.0 umoŕňuje souběŕný provoz aŕ 20 procesů DBW. Proto DBW bývá značen jako DBWn, kde n značí číslo procesu. Kdyŕ se ve vyrovnávací paměti nacházejí data, která jsou pozměněná následkem provedeného pŘíkazem `UPDATE`, tak je buffer označován jako dirty (dirty buffer). Zápis dat z dirty bufferu na disk je spuštěn událostí

zvanou checkpoint. Tyto události umožňují udržovat paměť volnou pro další buffery. Checkpoint může být buď plný (full), jenž spustil zapisování na disk ze všech dirty bufferů, nebo částečný (incremental), který spouští ukládání jen částí některých dirty bufferů. Plný checkpoint vyžaduje velké množství výkonu, a proto je spouštěn jen při vypínání databáze nebo manuálně na příkaz od administrátora. Checkpointy jsou řízeny procesem Checkpoint process (CKPT). CKPT spouští checkpointy v pravidelných intervalech. Kromě spouštění zápisu dat z paměti na disk se tento proces stará o udržování konzistence redo logů [27].

Oracle využívá paměťový prostor pro různé vyrovnávací paměti. Množina těchto vyrovnávacích pamětí se nazývá System Global Area (SGA). Vyrovnávací paměti v SGA jsou sdílené mezi uživateli, kteří přistupují k databázi. Zmíněnou vyrovnávací pamětí, jenž využívá proces DBW, se v Oracle konkrétně nazývá database buffer cache. Využívání tohoto sdíleného bufferu je výhodné, neboť když ke stejným datům přistupuje více uživatelů, tak tato data zůstanou v bufferu po skončení akcí prováděných prvním uživatelem, který musí počkat na natažení dat do vyrovnávací paměti z disku. Ostatní uživatelé pak rovnou přistupují k datům uložených v paměti po předchozím uživateli, což zvyšuje rychlost prováděných úkonů. Jakákoliv data však nemohou zůstat ve vyrovnávací paměti nekonečně dlouho. Pro určení, která data zůstanou v paměti déle, využívá Oracle algoritmus Least Recently Used (LRU), umožňujícího zredukovat množství přístupu na disk. LRU ponechává nejdéle v paměti data, která jsou nejvíce používaná [27].

Dalším typem vyrovnávací paměti je log buffer. Pokud je proveden commit nějaké změny v databázi, právě do této paměti se první zapíšou provedené změny. Informace z log bufferu jsou pak procesem LGWR zapisovány do redo logů (redo logy jsou popsány v kapitole 3.3) [27].

Kromě database buffer cache je v SGA ještě další vyrovnávací paměť. Jedná se o paměť shared pool, která se skládá ze čtyř částí (library cache, data dictionary cache, PL/SQL area, new result cache). Oracle jádro při každém dotazu na data musí zkontrolovat několik věcí. Jestli je příkaz syntakticky správně, jestli hledaná tabulka a v ní hledané sloupce existují nebo jestli uživatel, který příkaz zadal, má k datům oprávnění. Navíc je potřeba SQL příkaz přeložit na strojový kód. Tyto kroky se nazývají parsing. Právě shared pool se stará o urychlení těchto kroků tím, že paměť shared pool uchovává výsledky těchto kroků pro další podobné příkazy nebo dotazy na databázi [27].

Databázové datové soubory (data files) jsou fyzické soubory na disku pro ukládání téměř všech Oracle dat a jsou pro fungování databáze kriticky důležité. Jejich velikost se odvíjí přímo od množství dat, která jsou v nich uložena. Pro logické ukládání dat slouží ukládací prostor zvaný tablespaces. V tablespace jsou uloženy obvykle tabulky. Každý tablespace je přiřazený k jednomu nebo více datovým souborům. Pod textem je výstup SQL*Plus, který ukazuje v prvním sloupci všechny tablespaces a ve druhém sloupci všechny datové soubory (data files), ke kterým jsou přiřazeny [4]. Jak ukazuje obrázek 1 pod textem, každý datový soubor má v tomto případě jen jeden tablespace.

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

TABLESPACE_NAME     FILE_NAME
------------------------------   ------------------------
----------------------
SYSAUX    /opt/oracle/oradata/oracle_db1/sysaux01.dbf
SYSTEM    /opt/oracle/oradata/oracle_db1/system01.dbf
UNDOTBS1  /opt/oracle/oradata/oracle_db1/undotbs01.dbf
USERS     /opt/oracle/oradata/oracle_db1/users01.dbf
```

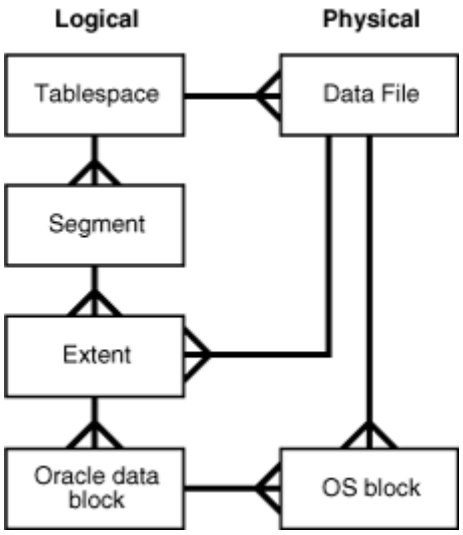
Obrázek 1: výstup z SQL*Plus zobrazující datové soubory a tablespaces - zdroj: vlastní zpracování

Velikost tablespace je definována relativně vzhledem k přiřazenému datovému souboru. Tablespace seskupuje logicky podobné segmenty. Ve výchozím nastavení má Oracle databáze minimálně tři tablespaces (SYSTEM, SYSAUX, TEMP) [4].

- SYSTEM - Ukládá datový slovník (data dictionary) a PL/SQL kód.
- SYSAUX - Uchovává segmenty používané pro různé funkce (Online Analytical Processing, Automatic Workload Repository).
- TEMP - Dočasný tablespace pro speciální operace, např. v případě dotazů na data, která jsou příliš velká na zpracování v operační paměti a je nutné využít disk, tak se využívá tento prostor.

Oracle rozděluje databázi na několik logických jednotek (obrázek 2). Toto rozdělení umožňuje efektivnější ukládání a získávání dat z databáze. Největšími jednotkami na než se databáze dělí jsou již zmíněné tablespaces. Tablespace se pak dělí

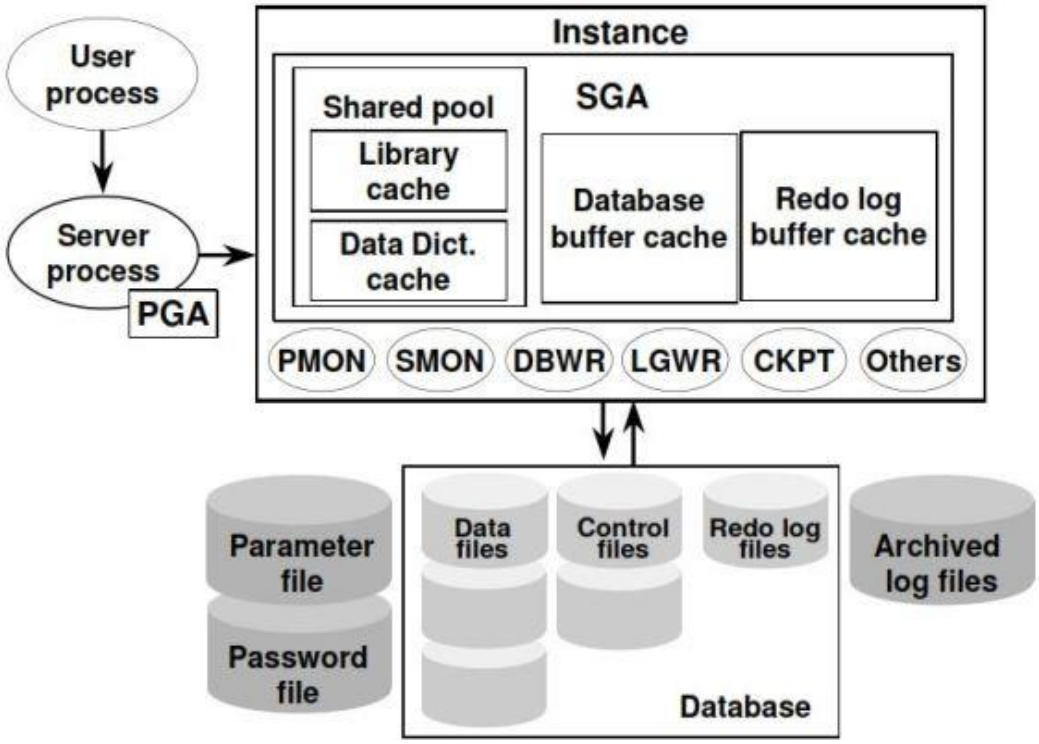
segmenty. Tyto logické jednotky uchovávají specifický typ datové struktury jako jsou např. konkrétní tabulky nebo indexy. Segment se skládá z jednotek zvaných extent, který se dále dělí na nejmenší jednotku zvanou datový blok [4].



Obrázek 2: fyzické a logické struktura databáze - zdroj: [36]

Druhou hlavní paměťovou oblastí Oracle architektury je program global area (PGA, obrázek 3). PGA není sdílenou pamětí, a proto se vytváří pro každého uživatele, který se připojí k databázi. Pokaždé, když uživatel spustí aplikaci přistupující k datům v databázi, Oracle spustí uživatelský proces, který je vyžadován k vytvoření spojení s databázovou instancí. Tento proces může běžet na stroji uživatele nebo na aplikačním serveru. Poté, co proces umožní uživateli se připojit, vznikne mezi uživatelem a databázovou instancí spojení (session) [27].

Po vytvoření session se pro každého uživatele spustí serverový proces na obsluhujícím serveru. Tento proces se stará o plnění úkonů, které umožňují komunikovat uživateli s databází (zprostředkovává komunikaci mezi uživatelem a Oracle instancí). Třetí částí je zmíněná PGA, do které se ukládají informaci o session. PGA se vytváří spolu se serverovým procesem [27]. Popsanou architekturu Oracle databáze shrnuje obrázek 3.



Obrázek 3: architektura Oracle instance a databáze - zdroj: [37]

3.3 Redo logy

Před podrobnějším popisem Data Guardu je potřeba uvést pojem redo. Redo log je jeden z nejdůležitějších pojmů pro recovery operace. Tyto záznamy uchovávají všechny potřebné informace o akcích provedených v databázi, aby v případě selhání databáze nebo ztráty dat mohly být tyto provedené změny zrekonstruovány. Pokaždé, když uživatel potvrdí transakci, redo informace, obsahující popis transakce, se přepíšou z vyrovnávací paměti (Redo log buffer) do online redo logů a po naplnění souboru s redo informacemi dojde k jeho archivaci do archivních redo logů (archived redo logs) [2].

Redo informace se uchovávají ve dvou či více redo logech. Online redo logy jsou minimálně dva z toho důvodu, aby byl alespoň jeden soubor dostupný, když je prováděna archivace, která během své činnosti omezí přístup k redo záznamu. Protože není způsob, jak obnovit ztracené redo logy, je proto potřeba je archivovat. Po naplnění jednoho redo logu se automaticky začne plnit další. Pro výběr dalšího logu používá Oracle pořadové číslo (redo logs sequence number). Toto číslo se ukládá do redo logů hned po použití. Po naplnění posledního redo logu se automaticky opět začne plnit první redo log. V případě že nechceme ztratit informace z redo logů, je tedy nutné soubory

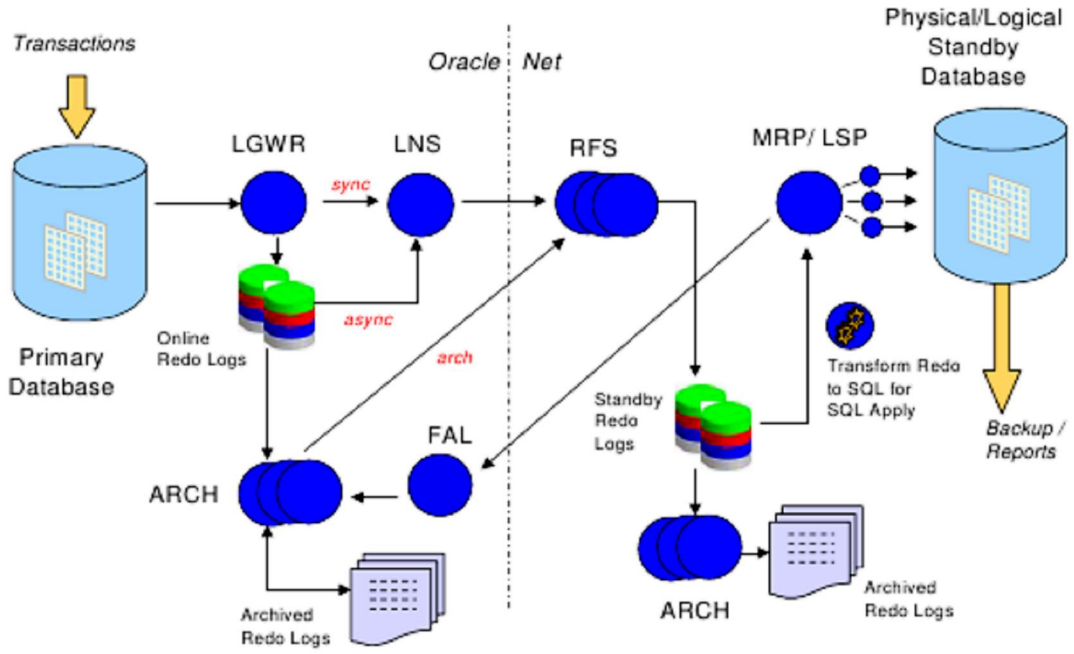
archivovat, než dojde k jejich přepsání. Redo logy, které se nacházejí v jedné instanci, se nazývají redo vlákna. (redo threads) [2].

3.4 Architektura Oracle Data Guard

První podnět, který spouští celý proces, je přijetí potvrzení nějaké transakce na primární databázi. V databázi se spustí proces Log Writer (LGWR), jenž reaguje při každém provedení změny v databázi. LGWR je proces běžící na pozadí. Proces řídí vyrovnávací paměť, ve které jsou zaznamenávány provedené transakce. LGWR pak z paměti zapisuje redo informace do online redo logů a přiřadí jim číslo pro identifikaci transakce [9, 27].

Další na řadu přichází Archiver proces (ARCH), který archivuje redo logy. Log Writer dále využívá procesy, skrývající se pod názvem LogWriter Network Server (LNS), k synchronnímu i asynchronnímu přenosu redo logů do úložiště standby databáze. Přenos redo logů probíhá v daném pořadí tak, aby byla zajištěna ochrana logů proti přerušení přenosu. V případě přenosu procesem Archiver dochází k posílání logů do standby databáze přímo bez využití dalších prostředníků. Dalším z procesů na primární databázi je Fetch Archive Log (FAL). FAL proces poskytuje mechanismus klient-server, který se stará o posílání archivovaných redo logů z primární do standby databáze. Využívá se v případech, kdy dojde k výpadku spojení mezi primární a standby databází pro automatické vyřešení výpadku a obnovení synchronizace [9, 18].

Do standby databáze probíhá přijímání redo logů z produkční databáze prostřednictvím procesu Remote File Server (RFS). Následně je prováděna aplikace redo logů pomocí Managed Recovery Process (MRP), který se používá u fyzické standby databáze. V případě logické standby databáze se jedná o proces Logical Standby Process (LSP) [9]. Celou architekturu graficky zobrazuje obrázek 4.



Obrázek 4: architektura Oracle Data Guard - zdroj: [9]

3.5 Změna rolí

Jak bylo již řečeno ODG konfigurace je tvořena typy databází. Primární a standby databáze představují role, ve kterých databáze fungují. Data Guard nabízí dva způsoby, jak role měnit. Jedná se o switchover a failover [3, 29].

3.5.1 Switchover

V jednoduché ODG konfiguraci je jedna primární databáze a jedna standby databáze. V případě, že je spuštěn switchover, dojde k obrácení rolí. Ze standby se stane primární a primární převezme úlohu standby databáze. Samozřejmě ihned po provedení switchoveru dochází ke změně směru posílání redo dat opačným směrem. V případě nasazení vícero standby databází lze switchover provést mezi primární databází a jakoukoliv standby databází v konfiguraci. V takovém případě nová primární databáze normálně pokračuje v rozesílání redo informací mezi všechny standby databáze [3, 29].

Switchover zaručuje, že nedojde ke ztrátě dat. Switchover je proto ideální pro plánované odstávky primární databáze, jelikož primární databáze je nedostupná jen po dobu jeho trvání.

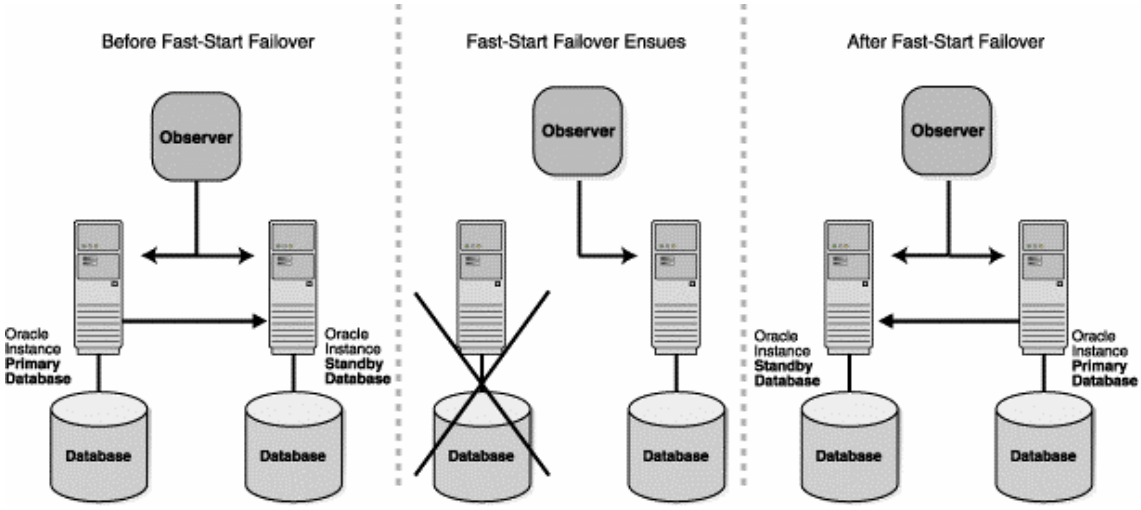
3.5.2 Failover

Jako v předchozím případě se jedná o operaci vykonávající změnu rolí. K přepnutí rolí však dochází z důvodu selhání primární databáze. V případě, že dojde k failoveru, dojde k přepnutí na novou primární databázi bez možnosti návratu na původní konfiguraci bez zásahu administrátora. Proto je výhodné při využívání Standby technologie jako ochrany proti selhání databáze využít technologii databázového Flashbacku. V případě failoveru s využitím Flashbacku databáze lze vrátit ODG konfiguraci do stavu před failoverem. V případě selhání primární databáze může být failover vykonán manuálně, což vykonává administrátor nebo automaticky (Fast-start failover) [7, 29].

Automatický failover může být zapnut jen v konfiguraci vytvořené pomocí Data Guard Brokeru. Pro automatický failover je potřeba zapnout proces Observer. Tento proces běží na serveru odděleném od primární a standby databáze, odkud nepřetržitě sleduje primární databázi. V případě, že pro Observer nebude primární databáze dostupná a po předem určené době nedojde k obnově dostupnosti, inicializuje se automatický failover a dojde k přepnutí role na standby databázi [3, 29].

V základním nastavení je několik událostí, které mohou failover vyvolat. Jedna z možností je ztráta spojení mezi procesem Observer a primární databází zaviněná síťovým selháním, což způsobí, že primární databáze je nedostupná. Dále patří mezi příčiny failoveru selhání databázové instance, poškození řídícího souboru (control file), nedostupnost datových souborů primární databáze, provedení uzavření databáze s volbou ABORT (SHUTDOWN ABORT) a porušení adresářové struktury datových objektů [23, 29].

Dodatečně je možné nastavit další možnosti inicializující failover, které se týkají redo záznamů. Jednak to je situace, kdy proces LGWR není schopen zapisovat data do redo logů z důvodu I/O chyby, a situace, ve které proces Archiver nemůže archivovat redo logy, jelikož ukládací medium není dostupné nebo na něm není dostatek vyhrazeného místa pro logy [2, 29]. Automatický failover zachycuje obrázek 5.



Obrázek 5: automatický failover - zdroj:[9]

4 Oracle standby databáze

4.1 Standby technologie

Oracle databáze zaznamenává všechny změny, které v ní nastanou, a ukládá je do změněných redo logů. ODG udržuje standby databáze synchronní s primární databází pomocí informací v těchto redo záznamech. Standby databáze se vytváří ze zálohy, kopírováním nebo klonováním z primární databáze na jiný server [2, 3].

Primární a standby databáze mohou existovat i na jednom serveru. Ale toto řešení by mělo být použito pouze pro testovací účely, jelikož by tím zmizela většina výhod, které Standby technologie přináší. V reálném nasazení je tedy žádoucí, aby databáze byly nasazeny v různých datových centrech v různých lokalitách. Po vytvoření standby databáze je pak pomocí ODG konfigurace zapnut přenos redo logů z primární na standby databázi a jejich aplikace na standby databázi, což umožňuje synchronizaci databází [2, 3, 30].

4.2 Aktualizace standby databáze

ODG automaticky posílá redo informace z primární na standby databázi, kde dochází k jejich aplikaci. Proces přenosu a proces aplikace redo logů na standby databázi probíhají nezávisle na sobě. ODG nabízí dvě možnosti aktualizace záložní databáze. Jedná se o služby Redo Apply a SQL Apply. Výběr konkrétní služby pak rozhoduje o podobě vnitřní struktury, tzn. zda se jedná o fyzickou nebo logickou standby databázi [3].

Jelikož základní cíl obou služeb je stejný, tedy ochrana dat pomocí udržování synchronní záložní databáze, tak mají několik společných charakteristik. Po přijetí redo logů standby databází probíhá kontrola fyzického poškození, aby se zabránilo jeho promítnutí do databáze. Ve výchozím nastavení standby databáze přijímá redo data a zapisuje je do vlastních standby redo logů a následně probíhá jejich archivace. Až poté dochází ve standby databázi k její aktualizaci pomocí obdržených dat [3].

ODG umožňuje celý proces urychlit pomocí funkce real time apply. Díky této funkci nedochází ke zpoždění zapříčiněného archivací a aktualizace probíhá hned po zápisu obdržených dat do redo logů. Dalším společným nastavením je čas zpoždění (Delay time). Tento parametr určuje, s jakým zpožděním se mají redo data aplikovat do

standby databáze. Toto nastavení může být užitečné v případě lidské chyby, kdy čas navíc zabrání přenosu chyby do standby databáze [3].

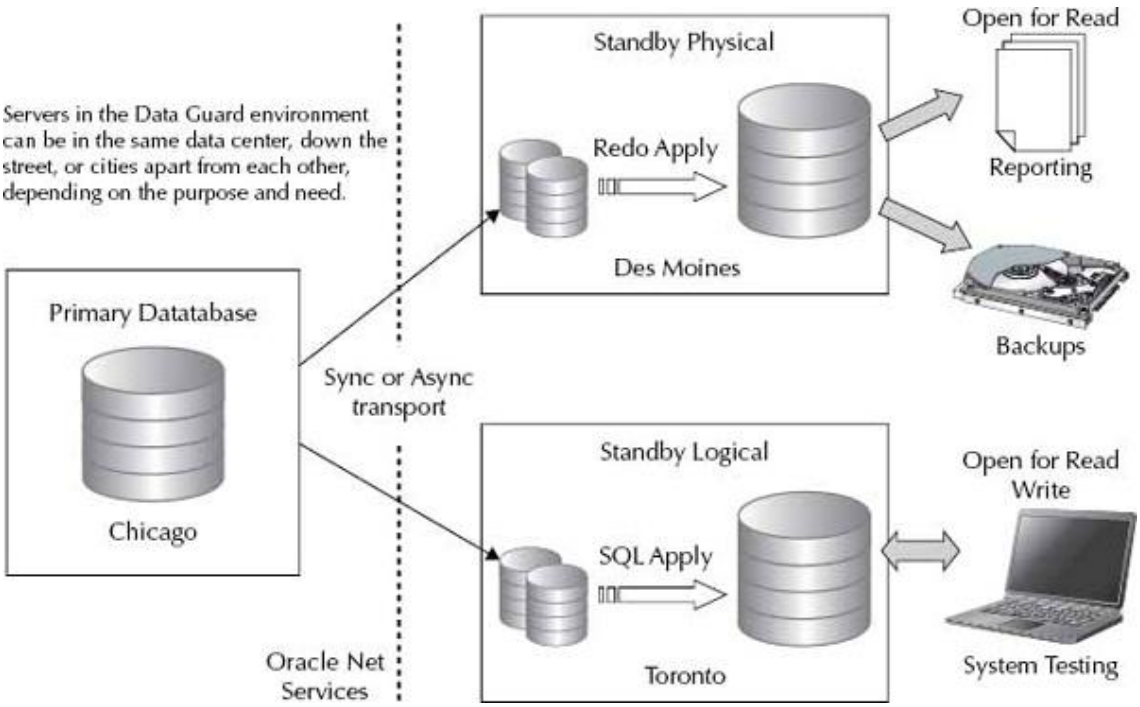
4.2.1 Redo Apply

Redo Apply je služba, ve které jsou redo data získaná z primární databáze aplikovaná do fyzické standby databáze. Proces udržuje standby databázi jako přesnou kopii primární databáze. Ve výchozím stavu Redo Apply běží jako paralelní procesy. Tyto procesy jsou po celou dobu kontrolovány pomocí Managed Recovery Process (MRP) procesu, který se stará o aktualizaci databáze pomocí redo informací [3].

4.2.2 SQL Apply

Druhým typem aktualizace standby databáze je SQL Apply. Jedná se o proces, který transformuje standby redo logy do DML příkazů (insert, update, delete). Následně jsou tyto SQL příkazy vykonávány ve standby databázi. Proto je logická standby databáze otevřena pro čtení i pro zápis [3].

Obrázek 6 znázorňuje fungování obou zmíněných služeb.



Obrázek 6: apply služby na fyzické a logické standby databázích - zdroj: [39]

4.3 Fyzická standby databáze

Tento typ standby představuje fyzicky identickou kopii primární databáze. Fyzická standby obsahuje jak stejná data, tak i stejné ostatní databázové objekty. Databáze je udržována synchronní s primární databází prostřednictvím služby Redo Apply. Databáze podporuje stejné datové typy i operace jako primární databáze. Fyzická standby může být otevřená v režimu pouze pro čtení (read only), pouze pokud právě nejsou aplikovány přijaté redo logy z primární databáze. Oracle zde nabízí vylepšení fyzické standby databáze v podobě Active Oracle Data Guard. Active ODG umožňuje fyzické standby databázi být v režimu read-only i v průběhu aplikování redo logů [2,3].

4.3.1 Active Oracle Data Guard

Fyzická standby databáze v Enterprise edici ani v žádném jiném vydání neumožňuje provádění čtení dat požadované uživateli a zároveň vykonávat update databáze z přijatých redo logů. V organizacích, kde databázi využívají desítky nebo dokonce stovky uživatelů, by pak nebylo možné fyzickou standby databázi využívat ke snížení zátěže kladené na primární databázi, protože by téměř veškerý čas zabralo nahrávání redo informací do databáze. Oracle jako řešení tohoto problému nabízí produkt Active Data Guard, který umožňuje být neustále v režimu čtení i během aktualizace standby databáze [2, 3].

Výhody:

- Při využití Active Data Guard lze standby databázi využít pro snížení zátěže na primární databázi
- Architektura zajišťující automatizovaný chod a stejná struktura primární a standby databáze usnadňují administrativu
- Přidává vyšší výkon ve srovnání s logickou standby databází
- Podpora všech datových typů, objektů, a DDL příkazů, které využívá primární databáze
- Redo Apply nabízí silnou kontrolu narušení dat a slouží i jako preventivní mechanismus

- V případě zapnutí real-time apply je standby databáze synchronizována s primární téměř okamžitě a tedy umožňuje přijímat dotazy, na které by jinak musela reagovat primární databáze

Fyzická standby databáze s Active Data Guardem velmi dobře řeší oblasti vysoké dostupnosti a disaster recovery. Za zmínku stojí upozornit na extra poplatek nad rámec licence Oracle databáze v Enterprise edici [31].

Nevýhody:

- Příplatek za Active Data Guard
- Active Data Guard pouze k licenci Enterprise edice

4.4 Logická standby databáze

Logickou standby databázi nabízí Oracle od verze 9i. Od fyzické standby databáze se liší především způsobem aktualizace standby databáze. Tento typ standby databáze využívá služby SQL Apply, která aktualizuje databázi pomocí SQL příkazů a umožňuje standby databázi být otevřenou pro čtení i zápis. Díky tomuto je možné nad databází provádět dotazy nebo získávat reporty nad rámec primární databáze. Dále lze provádět manipulaci s daty v tabulkách, které nejsou právě aktualizovány pomocí SQL Apply [3].

Také to umožňuje vytvářet úplně nové databázové objekty, které na primární databázi nejsou a ani existovat nemusí. Dokonce nemusí obsahovat všechna data z produkční databáze, anebo tam mohou být některá navíc. Samozřejmě je pak dále možné provádět s těmito daty jakoukoli manipulaci. Výsledkem je, že standby databáze má stejné (nebo alespoň jejich část) logické informace jako primární databáze (má stejná data jako primární databáze), ale rozdílnou fyzickou strukturu na disku. Z pochopitelných důvodů data, která tvoří zálohu z primární databáze, měnit nelze. Logická standby databáze umožňuje přidávat patche, nové releasy nebo provádět údržbu databáze s minimální dobou přerušení chodu databáze [3, 17].

V porovnání s Redo Apply, SQL Apply spotřebovává vyšší množství výkonu. To může zapříčinit vyšší množství vzniklých závad, a na rozdíl od fyzické standby databáze, logická nepodporuje všechny datové typy jako produkční databáze, což způsobuje, že standby databáze nemusí vždy obsahovat všechna data z primární

databáze. Proto logická standby databáze zajisté zvyšuje administrativní náročnost než fyzická standby databáze [3, 17].

Nepodporované typy logické standby databáze (Oracle 12c):
BFILE
ROWID,UROWID
Collection
Objekty s hnízděnými tabulkami (nested tables)
Identity Column
prostorové typy: MSSYS.SDO_GEORASTER, MDSYS.SDO_TOPO_GEOMETRY

Výhody:

- Není třeba odstávka databáze během provádění údržby nebo vylepšení Oracle softwaru
- Při velkých nárocích na reporting možnost přenesení zátěže z produkční databáze prostřednictvím přídavných objektů
- Možnost vytvořit ve standby databázi odlišnou strukturu od primární databáze
- Standby databáze je stále otevřená pro čtení/zápis, a tedy umožňuje přidávat tabulky, indexy, pohledy, atd.
- Umožňuje uchovávat data, která neexistují na primární databázi

U logické standby databáze stojí za to zdůraznit její nevýhody spojené s možností vytvořit odlišnou strukturu od primární databáze.

Nevýhody:

- Nepodporuje všechny datové typy
- Zvýšené požadavky na administraci

4.5 Snapshot standby databáze

Snapshot databáze slouží především pro testovací účely. Principiálně je snapshot databáze speciální verze fyzické databáze. V Data Guardu ji lze nalézt od verze 11g. Na rozdíl od předchozích dvou typů standby databází, snapshot hned po přijetí redo logy neaplikuje, ale uchovává si je. Redo data jsou použita až tehdy, když je databáze

přetvářena zpátky na fyzickou standby a všechny změny provedené na snapshot standby databázi jsou od jejího vytvoření smazané [3, 31].

Výhodou je, že poskytuje přesnou kopii primární databáze. Toto lze využít pro účely vývoje a testování, neboť data v primární databázi zůstanou netknutá. Vytváří se přeměnou z fyzické standby databáze. Po zrušení všech nežádaných změn provedených ve snapshot databázi se jednoduše provede aktualizace pomocí přijatých redo logů prostřednictvím přeměny na fyzickou standby databázi. Přeměna mezi snapshot a fyzickou standby databází může být prováděna během provozu neomezeně[3,31].

Výhody:

- Testování bez ohrožení produkční databáze
- Otevřena pro čtení a zápis

Jaký typ standby databáze je nejlepší, nelze jednoznačně určit. Volba typu záleží na konkrétních požadavcích a potřebách. Např. v situacích, kdy produkční databáze je v kritickém stavu, má časté výpadky a hrozí ztráta dat, tak prvotním cílem bude zajištění dostupnosti databáze a ochrana dat. Jasnou volbou je zde fyzická standby databáze, která zajistí ochranu všech dat primární databáze a jejich dostupnost, v případě selhání primární databáze [2, 3].

Jiná vzniklá situace je, když administrátor dostane hlášení, že kvůli pomalému chodu aplikací dochází ke snížení produktivity práce na jednom z oddělení. Jako hlavní příčina zpomalení aplikací je identifikován zvýšený počet reportů z dalšího oddělení, vzniklý z důvodu blížící se uzávěrky. Tady je rozhodovaní o nasazení konkrétního typu více komplexnější. Snapshot databáze nám nepomůže, neboť neobsahuje aktuální data. Do výběru tedy přichází fyzická standby s Active ODG nebo logická standby databáze. Pro finální řešení je zapotřebí vložit do rozhodování další proměnné jako jsou vlastní náklady v případě Active ODG nebo při nasazení logické standby databáze možnost přidat dodatečné objekty ale na úkor nedodržení její stejné struktury, což může být v rozporu s bezpečnostní politikou dané organizace [2, 3].

Tyto scénáře popisují případ, ve kterém jsou v ODG konfiguraci pouze dvě databáze. Ve skutečném světě není potřeba si vybírat. Stačí do ODG konfigurace přidat databázi požadovaného typu standby databáze [2, 3].

4.6 Reżimy standby databáze

Kromě ochranných režimů, které se starají především o způsob přenosu dat, tak lze nastavit recovery módy.

4.6.1 Manuální recovery režim

Oracle označuje tento režim jako manual recovery mode. Standby databáze v tomto režimu nepodporuje automatizované postupy ODG architektury. Redo data nejsou automaticky posílána na záložní databázový server. Manuální režim umožňuje využít pro přenos redo logů vlastní skript pro získání požadované funkcionality [41].

4.6.2 Automatický recovery režim

Tento režim nese anglické označení managed recovery mode. Jeho využití se nachází především v případě, je-li hlavní účel standby databáze ochrana proti narušení konzistence nebo ztrátě dat. V tomto módu primární databáze posílá redo logy na úložnou jednotku standby databáze, následně je standby databáze aplikuje a to vše zcela automaticky [41].

4.6.3 Read only režim

Read only režim (Read only mode) je vhodný, pokud standby databáze slouží primárně k získávání reportů. Bez nasazení Active Data Guard je fyzická standby databáze buď v recovery režimu, během něhož dochází k aplikaci redo logů nebo v režimu read only, kdy je dostupná pro uživatelské dotazy. Při nastavení tohoto módu standby databáze neaplikuje redo logy a je neustále otevřena pro čtení. Přijaté redo logy jsou databází uchovávány a přepnutím na automatický recovery mód lze pomocí nich kdykoli aktualizovat standby databázi [41].

4.7 Data Guard ochranné režimy

Výběr typu standby databáze je dále rozšířen o režimy, ve kterých primární a standby databáze operují. Pomocí těchto režimů se lze přiblížit požadovanému poměru mezi ochranou a výkonem. Standby databáze funguje v jednom ze tří režimů. Základní

charakteristikou je typ přenosu redo logů, od nějhož se následně odvíjejí další vlastnosti těchto módů. Při synchronním přenosu s čekáním na potvrzení ze standby databáze může docházet ke zpomalení primární databáze z důvodu dlouhého čekání. Jednotlivá čekání na odpověď se pak mohou začít hromadit, což může vyústit v nedostupnost databáze či ztrátu síťového spojení [30].

4.7.1 Maximální ochrana

Cílem módu maximální ochrany (Maximum Protection) je zajištění nejvyšší ochrany dat primární databáze. Pro zajištění vysoké ochrany je zapotřebí synchronní přenos redo informací mezi databázemi. Primární databáze před každým potvrzením změny čeká na zprávu alespoň od jedné standby databáze v ODG konfiguraci, že redo data úspěšně přijala a podle nich provedla aktualizaci (tedy, že jsou dostupné alespoň na disku jedné standby databáze). Pokud nastane situace, kdy standby databáze selže nebo dojde k přerušení jejího spojení s primární databází, tak následkem toho pak primární databáze nedostane odpověď na odeslaná data, proces LGWR nepotvrdí transakci a následně dojde k zastavení primární databáze, eventuelně může dojít i k ukončení databázové instance. Tímto dojde k zabránění vytvoření změn v primární databázi, které by nebyly promítnuty alespoň do jedné standby databáze. Díky těmto vlastnostem je zaručena kompletní ochrana dat i v případech výpadku spojení standby databáze a následně selhání primární databáze [2, 3].

Při každém provedení změny uživatelem je tato změna zaznamenána do redo logů. Dokud informace nejsou uloženy v redo logu, uživatel nedostane potvrzený commit svého příkazu. Tato vlastnost Oracle databáze spolu s režimem maximální ochrany zajišťuje nulovou ztrátu dat v primární databázi. Jediná data, o která lze v tomto režimu přijít, jsou data, na které uživatel nedostal potvrzený commit. V tomto módu jsou to data, nacházející se v době výpadku v log bufferu [2,3].

Při nasazení módu je doporučeno mít pro plynulý provoz v ODG konfiguraci minimálně dvě standby databáze. Při nasazení jedné standby databáze by v případě výpadku standby databáze došlo k zastavení i primární databáze. K zpřístupnění primární databáze by došlo až po opravení chyby, která způsobila nedostupnost standby databáze [2,3].

V testované verzi Oracle 12.1.0.2.0 není možné tento režim využít s funkcí Fast-start failover. Od následujícího vydání Oracle 12c je již automatický failover dostupný ve všech ochranných režimech [30].

Ze své podstaty má synchronní přenos dopad na časovou prodlevu při provádění změn a dotazů na data. Operace režimu se dají urychlit pomocí rychlého síťového spojení s nízkou latencí.

4.7.2 Maximální výkon

Režim maximálního výkonu (Maximum Performance) je výchozí ochranný mód. Jak název napovídá, tento režim nabízí vyšší výkon Data Guard konfigurace na úkor ochrany dat. V tomto režimu LGWR proces využívá asynchronní přenos redo dat. Popřípadě lze v tomto módu namísto LGWR nakonfigurovat Archiver proces k posílání redo logů. Asynchronní přenos znamená, že po odeslání redo dat proces LGWR nikdy nečeká na zprávu ze standby databáze o úspěšném provedení aktualizace databáze, ale rovnou provede potvrzení změn. Primární databáze tedy není nijak zpomalována zápisem redo dat do standby databáze [2, 3, 30].

V případě selhání primární databáze, redo data, které systém nestihl odeslat na standby databázi, jsou ztracena. Pokud však síťové spojení mezi databázemi má dostatečnou propustnost a o přenos dat se stará proces LGWR, je počet ztracených informací velmi malý. Mezi ztracenými daty v případě výpadku jsou jako v případě režimu maximální ochrany data z paměti SGA v log bufferu. Navíc v závislosti na okolnostech nezálohovaná (v případě výpadku ztracená) data mohou být ve všech pamětových částech Oracle architektury včetně datových souborů (data files) [30].

Režim je vhodný pro šíření, kde výkon a dostupnost databáze je přednější než možná ztráta malého množství provedených transakcí nebo tam, kde dělí databáze velké vzdálenosti, které způsobují vyšší časové prodlevy komunikace databázových strojů [2].

4.7.3 Maximální dostupnost

Tento režim, který lze nalézt v dokumentaci Oracle pod názvem Maximum Availability, má jako prioritu zajištění dostupnosti primární databáze. Představuje střední cestu mezi režimy maximálního výkonu a maximální ochrany, protože umožňuje podle potřeby mezi nimi automaticky přepínat. Po zapnutí režimu funguje na základě synchronního

přenosu dat. Jak bylo zmíněno u režimu maximální ochrany, tento přístup nese riziko zpomalení nebo dokonce zastavení databáze [2, 3, 30].

V situacích, kdy dojde k chybě na straně standby databáze nebo přerušení spojení se standby databází, ale primární databáze je v pořádku, může být režim maximální ochrany spíše nevýhodný. Zde přichází režim vysoké dostupnosti s rozšiřujícími charakteristikami oproti režimu maximální ochrany. V případě ztráty synchronizace režim vysoké dostupnosti čeká uživatelem nastavený čas v sekundách, který je definovaný parametrem `NET_TIMEOUT`. Ve výchozím stavu je hodnota nastavena na 30 vteřin. Po vypršení časového limitu primární databáze přestane čekat na odpověď ze standby databáze a potvrzením dokončí prováděnou transakci (po vypršení limitu je zapnut asynchronní přenos). Pokud se později spojení se standby databází obnoví, provede se opětovná synchronizace mezi databázemi pomocí nashromážděných archivovaných redo logů v úložišti primární databáze (je navrácen synchronní přenos dat). Od verze Oracle 12.1.0.2.0 je dostupný přenos nazvaný FastSync. Ve verzi 12.1.0.2.0 funguje pouze v režimu maximální dostupnosti. Tento režim vychází ze synchronního přenosu dat s rozdílem, že primární databáze nečeká na potvrzení o úspěšném zápisu dat do standby databáze ale stačí ji pouze, že instance obsluhující standby databázi redo data úspěšně přijala. Tímto je docíleno vyšší rychlosti oproti synchronnímu přenosu [2, 3]. Tabulka 1 ukazuje hlavní rozdíly mezi jednotlivými ochrannými režimy.

Oracle Enterprise edice 12.1.0.2.0	Maximální ochrana	Maximální dostupnost	Maximální výkon
Režim přenosu	Synchronní	Synchronní, Asynchronní, FastSync	Asynchronní
Potencionální ztráta dat na primární databázi	Žádná	Podle právě aktivního režimu přenosu	Všechna neodeslaná data do okamžiku výpadku
Rychlost primární databáze	Malá	Malá/Velká	Velká
Kompatibilní s Fast-start failover	Ne	Ano	Ano

Tabulka 1: porovnání režimů standby databáze - zdroj: vlastní zpracování

4.8 Far Sync

Far Sync je další vylepšení, které přináší přídavný produkt Active Oracle Data Guard. ODG Far Sync je vzdálená instance, jíž lze nastavit jako další součást Data Guard konfigurace. Tato instance přijímá redo logy z produkční databáze a rozesílá je ostatním členům Data Guard konfigurace. Far Sync instance využívá ke svému fungování řídící soubor (control file). Zde jejich podobnost s databázovými prvky ODG konfigurace končí. Far Sync instanci nelze převést v databázi, nemůže uchovávat data, ani používat apply služby [2, 3, 40].

Far Sync využívá velmi málo hardwarových prostředků. Umožňuje provádět failover s nulovou ztrátou dat. Dotazy způsobující přetížení primární databáze Far Sync posílá na standby databáze. Všechny způsoby přenosu, jež využívá primární databáze, jsou dostupné i na Far Sync [2, 3, 40].

Mnoho ODG konfigurací využívá asynchronní přenos, a tím riskují ztrátu dat v situaci, kdy dojde k výpadku primární databáze a inicializuje se failover, především z důvodu zpomalení primární databáze vznikajícího synchronním přenosem. Far Sync je pro tyto případy skvělým pomocníkem. Totiž vytvořením Far Sync instance v blízkosti primární databáze lze snížit časovou prodlevu vzniklou čekáním na odezvu od standby databáze. Primární databáze nečeká na potvrzení od standby databáze, ale pouze na odpověď Far Sync instance. Pokud Far Sync instance přijme redo záznamy, tak i v případě výpadku primární databáze a spuštěné failoveru, Far Sync instance odešle redo logy na standby databázi a vyčká na její potvrzení a poté vykoná failover, aby tak zajistila maximální ochranu dat [2, 3, 40].

4.9 Další výhody Active Oracle Data Guard

Vyjma Far Sync a možnosti využít fyzickou standby databázi pro recovery operace i pro získávání dat současně poskytuje Active Data Guard další benefity. Mezi ně patří schopnost opravit fyzickou poruchu datových bloků. Tato činnost minimalizuje nutnost zásahu administrátora a uživatelům zajišťuje práci bez přerušení.

Další dovednost Active Data Guard se nazývá Rolling Upgrades. Tato funkčnost poskytuje vysokou dostupnost. Umožňuje fyzickou standby databázi převést dočasně na logickou databázi na dobu potřebnou k provedení upgradu primární databáze. Tímto

způsobem je možné zajistit nepřetržitou dostupnost databáze při provádění údržby nebo při aktualizaci na novou verzi [2, 3, 40].

4.10 Uživatelské rozhraní pro administraci Oracle Data Guard

Existují tři možnosti jak ovládat a spravovat ODG. Jednak je možné ovládat pomocí příkazového řádku prostřednictvím SQL*Plus nebo skrz Data Guard Broker command line interface (DGMGRL). A jednak lze využít grafické uživatelské rozhraní, které nabízí Enterprise Manager [3, 31].

4.10.1 SQL*Plus

SQL*Plus je hlavní rozhraní využívající příkazový řádek pro správu Oracle databáze. Umožňuje zapnout a vypnout databázi, vytvářet uživatelské účty a databázové objekty či vkládat data. Prvním krokem při používání SQL*Plus je připojení se k databázi. Pro provádění změn nutných v konfiguraci a spravování ODG je nutné přihlásit se s přístupovými právy role sysdba. Tento způsob představuje nejvíce flexibilní možnost ovládání ODG. Konfigurace je však velice zdlouhavá. Každou změnu provedenou v ODG konfiguraci je potřeba vykonat na každém stroji opakovaně [2, 3].

4.10.2 DGMGRL

Oracle Data Guard Broker je framework, který automatizuje a centralizuje tvorbu, správu a monitorování ODG konfigurace. Oproti SQL*Plus usnadňuje administraci ODG konfigurace [2, 3].

ODG Broker má svůj vlastní příkazový řádek (DGMGRL) a syntaxi. Umožňuje některé složitější operace jako switchover nebo failover provádět pomocí pouze jednoho příkazu. ODG Broker umožňuje centralizovanou správu. Při provedení změny v ODG konfiguraci pomocí DGMGRL, ODG Broker propaguje automaticky změny k ostatním členům ODG konfigurace. ODG Broker obsahuje důležité příkazy, bez nichž by nešlo plně využít Oracle Data Guard. Např. obsahuje příkaz pro zapnutí a vypnutí procesu Observer, bez kterého by nefungoval automatický failover. K plnému využívání ODG Brokeru je nutné se přihlásit pomocí uživatele SYS [2, 3].

4.10.3 Enterprise Manager

Oracle Enterprise Manager nabízí GUI pro ODG Broker. Díky Enterprise Manageru lze jednoduše spravovat ODG, provádět switchover nebo přidávat a mazat databáze z ODG konfigurace. Dále umožňuje přehledně sledovat hlavní informace a výkon konfigurace nebo stavy synchronizace databází [2, 3].

4.11 Výběr vhodného typu standby databáze

Před nasazením Oracle Data Guard je potřeba vybrat vhodný typ standby databáze a typ ochranného režimu pro účely informačního systému studijní agendy (IS/STAG). Potřeby IS/STAG byly konzultovány s vedoucím této práce a na základě domluvy byl vybrán vhodný typ standby databáze. Postup výběru byl následující.

Nejdůležitějším požadavkem na standby databázi je, aby zajišťovala nepřetržitou ochranu dat produkční databáze obsahující data informačního systému a zajišťovala k nim nepřetržitý přístup.

Na výběr jsou tři typy standby databází, které byly popsány na začátku této kapitoly. Standby databáze typu snapshot nepřichází v úvahu, protože poskytuje data produkční databáze pouze k času svého vytvoření a jejím účelem je převážně testování. Na výběr přicházejí v úvahu fyzická nebo logická standby databáze.

Rozdíly obou standby databází zobrazuje tabulka 2, která poslouží jako pomoc při rozhodování. První řádek udává apply služby. Apply služby představují rozhodnutí mezi vyšší rychlostí aktualizace standby databáze, kterou představuje volba Redo Apply oproti pomalejší možnosti SQL Apply (samozřejmě rozdíl apply služeb není jen v rychlosti, z pohledu rozhodování postačuje se na výběr dívat takto zjednodušeně). Z tohoto pohledu na apply služby je výhodnější zvolit Redo Apply. Další dvě položky hodnotí nároky na administrátory IS/STAG. I zde lépe vychází fyzická standby databáze. Nicméně administrativní náročnost není z pohledu CIV, který se stará o systém STAG, rozhodující parametr. Další jsou podporované datové typy. IS/STAG obsahuje datové typy, jenž logická standby databáze nepodporuje, a tudíž by bylo nutné data nepodporovaných datových typů ochraňovat jiným způsobem. IS/STAG obsahuje datový typ Collection, jenž patří mezi nepodporované datové typy (viz kapitola 4.4).

Další tři řádky tabulky 2 informují o možnostech využití standby databáze pro snížení zátěže produkční databáze. Na první pohled tyto řádky svědčí ve prospěch

logické standby. Na volbu je vůbec potřeba pohlížet v kontextu s potřebami IS/STAG. Produkční databáze IS/STAG běží v současnosti na dostatečně výkonném stroji, který zvládá všechny požadavky, jenž jsou na ně kladeny. Pro současné potřeby IS/STAG tudíž není potřeba využívat standby databázi pro odklon zátěže a tyto volby tedy nejsou rozhodující.

Poslední dva řádky tabulky 2 shrnují možnost přidat do standby další objekty, což následně způsobuje změnu struktury standby databáze. Po konzultaci s vedoucím této práce bylo zjištěno, že přidávání dodatečných objektů do standby není pro potřeby informačního systému STAG vyžadováno. Po zhodnocení obou typů standby databází vyšla celkově lépe fyzická standby databáze, která splňuje potřebu vysoké ochrany dat. Pro zamítnutí logické standby databáze hovoří především špatná podpora datových typů. Jako nejvhodnější typem standby databáze, který byl vybrán pro otestování Oracle Data Guard technologie pro účely IS/STAG, je tedy fyzická standby databáze.

Druhým rozhodnutím je určení nejvhodnějšího ochranného režimu, ve kterém ODG konfigurace se zvolenou fyzickou standby databází pobíží. Vybrán byl mód maximální dostupnosti, který čeká na potvrzení standby databáze, čímž zajišťuje nulovou ztrátu dat a zajišťuje i dostupnost primární databáze v případě výpadku standby databáze. I režim maximální ochrany dostatečně chrání data. Ten byl však zamítnut z důvodu nekompatibility s funkcí Fast-start failover.

Oracle EE 12.1.0.2.0	Fyzická standby databáze	Logická standby databáze
Využívaná služba pro aktualizace	Redo Apply	SQL Apply
Administrativa	Jednoduchá	Náročnější
Vytváření standby databáze	Jednoduché	Náročnější
Podporované datové typy	Všechny	Nepodporuje všechny datové typy
Přístup k datům ve standby	Pouze čtení	Čtení i zápis
Přístup k datům standby při probíhajících aktualizacích	Není	Pouze čtení
Možnost převzít zátěž primární databáze	Jen velmi omezeně	Ano
Struktura standby databáze	Vždy shodná s primární databází	Obvykle odlišná od primární databáze
Možnost přidat další objekty do standby databáze	Ne	Ano

Tabulka 2: porovnání standby databází - zdroj: vlastní zpracování

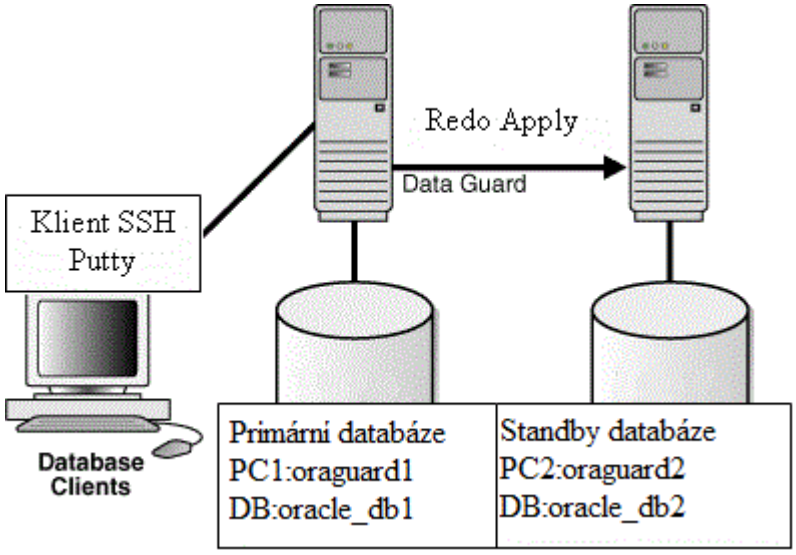
5 Implementace Oracle Data Guard

5.1 Prostředí pro nasazení Oracle Data Guard

Prvním krokem praktické části je implementace databáze v Enterprise edici verze 12.1.0.2.0. Implementace probíhala v prostředí Centra informatizace a výpočetní techniky (CIV), které zajišťuje provoz a rozvoj informačních technologií na půdě ZČU. K účelům diplomové práce byly centrem CIV poskytnuty dva virtuální stroje, které jsou součástí cloudových služeb ZČU. Na těchto virtuálních počítačích probíhala implementace a vyzkoušení databázové technologie Oracle.

Na virtuálních strojích byl nasazen operační systém linuxové distribuce Debian 9 stretch. Hardwarové komponenty těchto strojů byly procesor Intel Xeon 2,1 GHz, operační paměti o velikosti 4 GB a 20 GB diskového místa. Oracle verze 12.1.0.2.0 pro instalaci své databáze v Enterprise edici na virtuální stroj s linuxovým operačním systémem vyžaduje minimálně 1 GB RAM (doporučená velikost jsou 4GB) a 6,4 GB místa na disku.

Virtuální počítače nesly názvy oraguard1 a oraguard2. K přístupu ke strojům byly využity terminálové služby eryx, které poskytují distribuované výpočetní prostředí Orion na ZČU. K připojování ke službám eryx, a tím i k testovacím strojům byl využit program Putty, jenž zabezpečil spojení protokolem SSH. Obrázek 7 představuje prostředí virtuálních strojů s databázemi, k nimž se uživatel připojuje prostřednictvím SSH klienta Putty.



Obrázek 7: schéma virtuálních strojů a Data Guard - zdroj: [42], vlastní zpracování

5.2 Příprava operačního systému na instalaci databáze

Prvním krokem je příprava virtuálních počítačů k instalaci databázového softwaru. Tato část se skládala z několika kroků. První krok není spuštění samotné instalace. Protože instalovaná linuxová distribuce Debian nepatří mezi podporované operační systémy společností Oracle, je tedy nutné nejdříve prostředí Debianu pro instalaci připravit.

Debian sice není podporovaný operační systém, to ovšem neznamená, že by Oracle databáze na něj nešla instalovat. Na internetu existuje mnoho webů s průvodci a internetových fór, zabývajících se instalací Oracle produktů na nepodporované operační systémy.

Příprava Debianu se skládá z několika kroků [44]. Nejdříve je potřeba provést aktualizaci všech programových balíčků, které jsou instalované na systému na nejnovější verze a instalovat některé nové programy. Následujícími příkazy se spustí jejich stažení a poté instalace. První dva příkazy se starají o aktualizaci stávajícího softwaru. Další čtyři příkazy přidávají balíčky, jež Oracle databáze vyžaduje. Například třetí příkaz instaluje program Alien, který umožňuje konverzi mezi různými standardními linuxovými formáty, nebo například příkaz 5.4 přidává do systému program GNU Compiler Collection (GCC), což je sada překladačů pro různé jazyky [16, 44].

Pro budoucí konfigurace je výhodné změnit výchozí shell (příkaz 5.1).

```
root@oraguarrd1:~# apt update                                    (5.1)
root@oraguarrd1:~# apt upgrade                                   (5.2)
root@oraguarrd1:~# apt install alien binutils build-
essential                                                        (5.3)
root@oraguarrd1:~# apt install gcc-4.8-locales libstdc++-
4.8-dev lib32stdc++ -4.8-dev libc6-dev                           (5.4)
root@oraguarrd1:~# apt install libaio-dev libcap-dev
libpcap-dev libxau-dev libxcb1-dev libxi-dev
libxtst-devq                                                     (5.5)
root@oraguarrd1:~# apt install ksh sysstat xauth
unixodbc-dev                                                     (5.6)
root@oraguarrd1:~# ln -sf bash /bin/sh                           (5.7)
```

Protože se jedná o nepodporovaný operační systém, Oracle databázový instalátor očekává některá nastavení operačního systému, které neodpovídají tomu, jak je má Debian. Tato nastavení je tedy potřeba upravit. Jedná se o změnu umístění základních utilit. K vykonání změn je využit nástroj ln (příkazy 5.8, 5.9, 5.10), který vytváří symbolický odkaz na jiné umístění [5, 44].

```
root@oraguarrd1:~# ln -s /usr/bin/awk /bin/a                     (5.8)
root@oraguarrd1:~# ln -s /usr/bin/basename
/bin/basename                                                    (5.9)
root@oraguarrd1:~# ln -s /usr/bin/rpm /bin/rpm                   (5.10)
```

Kromě utilit je potřeba pozměnit i lokace knihoven, se kterými Oracle pracuje. K vykonání opět slouží nástroj ln, jak ukazují následující příkazy 5.11, 5.12, 5.13 [16, 44].

```
root@oraguarrd1:~# ln -s /lib/x86_64-linux-
gnu/libgcc_s.so.1 /lib64/                                        (5.11)
root@oraguarrd1:~# ln -s /usr/lib/x86_64-linux-
gnu/libstdc++.so.6 /lib64/                                       (5.12)
```

```
root@oraguarrd1:~# ln -s /usr/lib/x86_64-linux-gnu
/usr/lib64
```
(5.13)

Další zmĕnou umístĕní oproti oĉekávání Oracle jsou skripty spouštĕné pĜi startu systému. Oracle hledá tyto skripty v adresáĜi `/etc/rc.d`. Tudíž je potĜeba tento adresáĜ pĜipravit. VytvoĜení probíhá pomocí nástroje mkdir (pĜíkaz 5.14).

```
root@oraguarrd1:~# mkdir /etc/rc.d for i in 0 1 2 3
4 5 6 S ; do ln -s /etc/rc$i.d /etc/rc.d/rc$i.d;
done
```
(5.14)

Dále je potĜeba vytvoĜit uživatelské skupiny a nového uživatele, jehož prostĜednictvím se provede instalace Oracle softwaru a databáze, a který bude dále využíván pro implementaci a testování ODG. Vykonáním pĜíkazĤ 5.15, 5.16, 5.17 dojde k vytvoĜení skupin oinstall, dba a oper. PĜíkaz 5.18 vytvoĜí uživatele `oracle` s heslem `oracle` spolu s jeho domovským adresáĜem a nastavením shellu na `/bin/bash` [16, 44].

```
root@oraguarrd1:~# groupadd oinstall
```
(5.15)
```
root@oraguarrd1:~# groupadd dba
```
(5.16)
```
root@oraguarrd1:~# groupadd oper
```
(5.17)
```
root@oraguarrd1:~# useradd -r -m -g oinstall -G
dba,oper -d /home/oracle -s /bin/bash oracle
passwd oracle
```
(5.18)

Dále je vhodné vytvoĜit adresáĜovou strukturu pro instalaci databázového softwaru a prostĜednictví nástroje chown uživateli `oracle` pĜidĕlit práva skupiny dba k adresáĜĤm, které Oracle instalátor používá (pĜíkazy 5.19, 5.20, 5.21). AdresáĜ v pĜíkazu 5.21 je adresáĜ s instalaĉními soubory Oracle databáze [5, 44].

```
root@oraguarrd1:~# mkdir -p /opt/oracle/
```
(5.19)
```
root@oraguarrd1:~# chown -R oracle:dba /opt/oracle
```
(5.20)

```
root@oraguarrd1:~# chown -R oracle:dba                    (5.21)
/var/tmp/oracle
```

Poslední sadou příkazů se upravuje soubor limits.conf, který udržuje parametry zdrojů, rozdělené mezi uživatele a uživatelské skupiny a soubor sysctl.conf, umožňující prohlédnout si a měnit parametry jádra. Jednou z možností jak upravit soubory je pomocí nástroje echo. Použití tohoto nástroje k úpravě souborů ukazují následující příkazy 5.23 a 5.22 [5, 44].

```
root@oraguarrd1:~# echo "oracle soft nproc 2047" |
tee -a /etc/security/limits.conf
echo "oracle hard nproc 16384" | tee -a
/etc/security/limits.conf
echo "oracle soft nofile 1024" | tee -a
/etc/security/limits.conf
echo "oracle hard nofile 65536" | tee -a
/etc/security/limits.conf
echo "oracle soft stack 10240" | tee -a
/etc/security/limits.conf
echo "oracle hard stack 32768" | tee -a
/etc/security/limits.conf
echo "oracle soft memlock 3145728" | tee -a
/etc/security/limits.conf
echo "oracle hard memlock 3145728" | tee -a
/etc/security/limits.conf                                 (5.22)
```

```
root@oraguarrd1:~# echo "fs.aio-max-nr=1048576" |
tee -a /etc/sysctl.conf
echo "fs.file-max=6815744" | tee -a /etc/sysctl.conf
echo "panic_on_oops=1" | tee -a /etc/sysctl.conf
echo "net.ipv4.ip_local_port_range=9000 65500" | tee
-a /etc/sysctl.conf                                       (5.23)
```

Některé z těchto změn se projevují až po restartování počítače. Tudíž posledním krokem přípravy Debianu je provést restart systému.

5.3 Instalace Databáze

Samotnou instalaci lze rozdělit na dva kroky. Na instalaci Oracle databázového softwaru a databáze. Oracle umožňuje instalaci obou částí najednou, ale je lepší nejdříve instalovat binární soubory databázového softwaru, neboť už to je značně velký krok a až poté, co proběhne úspěšná instalace Oracle softwaru, tak spustit instalaci databáze.

Po spuštění souboru runInstaller v adresáři s instalačními soubory databáze se zobrazí klasický průvodce instalací. První krok nabízí možnost zadat emailovou adresu. Tento krok byl na počítači oraguard1 přeskočen. Další krok umožňuje zvolit zmiňovanou volbu instalace `Software Only` (instalace bez databáze). Dále pak přichází na výběr možnost instalovat instanci jako součást clusteru. Technologie RAC však není cílem této práce, a proto byla vybrána možnost `single instance`. Další krok nabízí možnost zvolit si i jiný jazyk než výchozí nastavená angličtina. Poté přichází na řadu výběr databázové edice. Na virtuálních počítačích byla zvolena Enterprise edice. Instalace proběhla do výchozího linuxového adresáře `/opt`, konkrétně do podadresáře `/oracle`, který byl vytvořen v předinstalační přípravě. Předposlední krok před zahájením instalace je nastavit systémové skupiny pro několik typů uživatelských aktivit. Při tomto kroku bylo na stroji oraguard1 ponecháno výchozí nastavení instalátoru. Před spuštěním instalace se ještě zobrazí okno s přehledem zvolených nastavení. Dalším krokem se spustí instalace. Před dovršením sta procent instalátor vybídne ke spuštění dvou skriptů, které je nutné spustit s oprávněním uživatele root. Po spuštění těchto skriptů je instalace kompletní [12].

Až po instalaci databázového softwaru byly na obou dvou počítačích vykonány výše uvedené kroky. Další postup byl proveden pouze na oraguard1, neboť vytvoření standby databáze probíhalo pomocí příkazového řádku RMAN. Tento postup je popsán dále v této práci.

Pro instalaci samotné databáze velmi dobře slouží databázový konfigurační asistent (DBCA). Tento průvodce se nachází v adresáři `/bin` v domovském adresáři instalovaného Oracle softwaru. Po spuštění DBCA průvodce hned nabízí požadovanou možnost vytvořit databázi. V dalším okně je nabízen výběr mezi jednoduchou

a pokročilou instalací. Při instalaci byla zvolena druhá možnost, která nabízí detailnější možnost konfigurace. V prvním kroku instalace s rozšířeným nastavení je volba šablony, jež např. udává strukturu datových souborů databáze. Pro ODG je postačující šablona instalace pro základní účel a transakční zpracování (general purpose and transactional processing). V další fázi je potřeba zadat název databáze. Pro počítač oraguard1 byl zvolen název databáze (GLOBAL_NAME) `oracle_db1` a SID `oracledb1`. Další rozhodnutí spočívá v tom, zda bude využíván Enterprise Manager (EM). Z důvodu nedostatku diskového prostoru nebyla možnost konfigurace EM při instalaci vybrána. Dále je vyžadováno heslo pro administrativní úkony. V dalším okně je potřeba vytvořit listener. Listener je proces, jenž běží na straně serveru a poslouchá příchozí databázové spojení. Volba listeneru se nachází v instalaci databáze od verze 12c a jeho přítomnost je důležitá pro proces vytváření databáze. Vytvořený listener na oraguard1 byl pojmenován LISTENER a poslouchá na portu číslo 1521 [12].

V dalším kroku se volí adresáře pro ukládání souborů. První volbou je adresář pro datové soubory. Adresář, který je potřeba zvolit pro funkčnost ODG, je Fast Recovery Area. Tento adresář určuje, kam se budou ukládat soubory záloh a recovery soubory. Při nevyužití volby Fast Recovery Area v instalátoru je možné adresář vytvořit později. Při nasazení ODG je Fast Recovery Area výchozím adresářem pro ukládání archivovaných redo logů a souborů Flashbacku. Při nasazení na virtuálních strojích byla využita i možnost nastavit maximální velikost tohoto adresáře a to na velikost 2 GB z důvodu úspory diskového místa. Zbytek nastavení v tomto kroku byl ponechán ve výchozím stavu. Další krok instalátoru umožňuje konfiguraci zabezpečení a řízení přístupu k datům. Tyto volby nejsou pro ODG potřeba, a proto při instalaci zůstaly nevyplněné.

Posledním krokem instalátoru je definování využívání paměti, inicializačních parametrů a jazyků. Velikost využívané paměti zde byla nastavena na 1 GB a znaková sada nastavena na `AL32UTF8`. Ostatní volby byly ponechány ve výchozím nastavení. V následujícím okně instalačního průvodce lze zvolit vytvoření databáze. Po této volbě se zobrazí přehled se vším zvoleným nastavením, a poté se již rozběhne proces vytváření databáze.

Po dokončení instalace je databáze automaticky spuštěna. Pro další konfiguraci je využíván konzolový klient SQL*Plus. Před spuštěním SQL*Plus je nutné zadat dva parametry `ORACLE_HOME` a `ORACLE_SID`. Bez nastavení těchto parametrů pokus

o přihlášení do SQL*plus skončí chybami SP2-0667 a SP20750. Konfigurace těchto parametrů na počítači oraguard1 byla provedena pomocí příkazů 5.24 a 5.25. Při využití těchto příkazů je nevýhodou, že je nutné nastavovat parametry při každém dalším přihlášení uživatele. Trvalým řešením je umístit tyto příkazy do souboru `.profile`, který se nachází v domácím adresáři uživatele (obsah souboru `.profile` počítače oraguard1 lze nalézt v příloze). Příkazu 5.26 zapne SQL*Plus bez připojení se k databázi. Příkaz 5.27 slouží k připojení se v roli sysdba. Pokud by databáze nebyla spuštěna, tak k jejímu nastartování slouží příkaz 5.28 [6].

```
oracle@oraguard1:~$ export oracle_home=/opt/oracle/
product/12.1.0/dbhome_1                                    (5.24)
oracle@oraguard1:~$ export oracle_sid=oracledb1            (5.25)

oracle@oraguard1:/opt/oracle/product/12.1.0/dbhome_
1/ bin$./sqlplus /nolog                                    (5.26)
SQL> conn / as sysba                                       (5.27)
SQL> startup                                               (5.28)
```

Před začátkem další podkapitoly je potřeba umět rozlišovat především dvě označení, která má každá databáze. Jedno z nich je `DB_UNIQUE_NAME`. Toto unikátní jméno databáze musí mít každá databáze jiné, aby ji bylo možné identifikovat. Tím spadá toto označení mezi jedno z mála věcí, ve kterých se primární a standby databáze odlišují. `DB_UNIQUE_NAME` je vytvořený ze jména `GLOBAL_NAME` zadávané při instalaci databáze. Druhým označením je `DB_NAME`, které musí být pro primární i fyzickou standby databázi stejné. Oba názvy databází lze zjistit využitím SQL*PLUS a zadáním příkazů 5.29 a 5.30 [3].

```
SQL> show parameter db_name                                (5.29)
SQL> show parameter db_unique_name                         (5.30)
```

Tabulka 3 zobrazuje přehled označení a identifikátorů testovacích databází, které byly použity při vytváření databází. Druhý stroj v tomto kroku zatím nemá žádná

označení. Před začátkem vytváření standby databáze je ale potřeba mít tyto označení připravena.

	Primární databáze	Standby databáze
DB_UNIQUE_NAME	oracle_db1	oracle_db2
DB_NAME	oracle_d	oracle_d
GLOBAL_NAME	oracle_db1	oracle_db2
SID	oracledb1	oracledb2

Tabulka 3: přehled označení databází - zdroj: vlastní zpracování

5.4 Vytvoření standby databáze

5.4.1 Vytvoření standby fyzické databáze - příprava primární databáze

Před samotným vytvořením standby databáze je potřeba připravit primární databázi. Většinu nastavení přípravných kroků lze provést prostřednictvím SQL*Plus s právy role sysdba. Nejdříve je potřeba zapnout vytváření plnohodnotných redo logů, prostřednictvím funkce `FORCE LOGGING` (příkaz 5.31), které jsou nutné pro recovery operace. Tímto dojde k vytváření redo logů se všemi potřebnými informacemi nutnými pro aktualizaci standby databáze. Je nutné brát na zřetel, že tato funkce sebou přináší vyšší výkonnostní nároky než výchozí funkce `NO LOGGING`, která tvoří pouze redo logy se základními informacemi. Dále ODG vyžaduje vlastní soubor pro redo logy, do kterých si ukládá redo data přijatá z primární databáze. Příkazem 5.32 lze vytvořit tyto logy s velikostí 50 MB. Tyto tzv. standby redo logy musí být stejně velké nebo větší než největší redo log primární databáze. Při přípravě primární databáze byly vytvořeny čtyři standby redo logy s velikostí 50 MB. Oracle doporučuje vytvářet redo logy na primární databázi a posléze je přenést do lokace standby databáze pro případné provedení operací switchover a failover. Přenesení těchto logů je docíleno samotným procesem vytváření, který je popsán níže [10, 11, 27].

```
SQL> alter database force logging;                    (5.31)
SQL> alter database add standby logfile size 50M;     (5.32)
```

Další parametry, které je nutné nastavit jsou LOG_ARCHIVE_CONFIG a LOG_ARCHIVE_DEST. První zmíněný parametr nastavuje dvojici primární a standby databází, mezi kterými budou proudit redo logy. Druhý parametr nastavuje přenos redo logů. Vykonáním příkazu 5.33 dojde k zapnutí přenosu mezi primární databází oracle_db1 a standby databází oracle_db2. Do příkazu se zadávají unikátní jména databází (DB_UNIQUE_NAME). Další příkaz 5.34 nastavuje mnoho atributů, polí naje SERVICE, který určuje název standby databáze v souboru TNSNAMES.ORA. Atribut ASYNC zařídí, že se po vytvoření standby databáze spustí asynchronní přenos v režimu maximálního výkonu. VALID_FOR je nepovinný atribut. Nicméně Oracle doporučuje jeho nastavení s hodnotami ONLINE_LOG_FILE a PRIMARY_DATABASE. Tímto je docíleno, že redo data budou ukládána v definované lokaci. Tedy, když je databáze v primární roli, tak online redo logy budou v jejím umístění i archivovány. Příkaz opět používá DB_UNIQUE_NAME. Pro správnou funkčnost se musí toto unikátní jméno databáze shodovat se jménem standby databáze v příkazu 5.33 [2, 13].

Dále se nastavuje automatické promítnutí změn do standby databáze (příkaz 5.35). Tento mechanismus je platný pouze pro fyzickou standby databázi (při používání Redo Apply). Poslední funkčností v tomto kroku, kterou je nutno zapnout, je archivace redo logů (příkaz 5.36). Databáze v tomto režimu zálohuje redo záznamy. Dokud nedojde k vytvoření zálohy redo logu, tak v tomto módu nemůže dojít k jeho přepsání [24, 25, 26].

```
SQL> alter system
log_archive_config='dg_config=(oracle_db1,oracle_db2
)';                                                    (5.33)
SQL> alter system set log_archive_dest_2=
'service=oracle_db2 async valid for=
(online_logfile,primary_role)
db_unique_name=oracle_db2;                             (5.34)
SQL> alter system set standby_file_managent=auto;      (5.35)
SQL> alter database archivelog;                        (5.36)
```

Krok, který není nutný pro vytváření standby databáze, ale je vhodné ho vykonat před vytvoření standby databáze, neboť se při vytváření zkopíruje do standby a tedy ho

není nutné vykonávat dvakrát, je zapnutí databázového Flashbacku (příkaz 5.37). Tato komplementární technologie k ODG je nutná pro funkci automatického failoveru [13].

```
SQL> alter database flashback on;                    (5.37)
```

Provedené změny databáze lze ověřit pomocí následující skupiny příkazů. Screenshoty terminálu s příkazy lze nalézt v příloze této práce.

```
SQL> select force_logging from v$database;           (5.38)
SQL> select * from v$log;                            (5.39)
SQL> select * from v$logfile;                        (5.40)
SQL> archive log list;                               (5.41)
```

Další postup přípravy již nevyžaduje příkazový řádek SQL*PLUS. V tomto kroku je potřeba upravit konfigurační soubory TNSNAMES.ORA a LISTENER.ORA. Soubor TNSNAMES.ORA definuje databázové adresy, které se využívají pro vytváření spojení databází (kromě adres soubor také obsahuje využívaný port a přenosový protokol, parametry databáze `SID` a `GLOBAL_DBNAME` a domovský adresář databázového softwaru). Proto je nutné pro fungující komunikaci databází, aby se v konfiguračním souboru nacházely informace o obou databázích. Screenshot upraveného TNSNAMES.ORA ze stroje oraguard1 lze nalézt v příloze této práce (obrázek 24) [2].

Druhým zmíněným souborem je LISTENER.ORA, sloužící procesu listener pro uchování parametrů. Nakonfigurovaný LISTENER.ORA ukazuje obrázek 22 v příloze této práce. Pro vytvoření těchto souborů Oracle nabízí programy Netca a Netmgr. První zmíněný však neumožňuje nakonfigurovat soubory požadovaným způsobem. Netmgr nabízí více možností konfigurace parametrů a lze ho využít pro vytvoření těchto souborů. Nicméně soubory lze upravit i manuálně. Pro projevení změn v souboru LISTENER.ORA je potřeba listener restartovat [2].

5.4.2 Vytvoření standby fyzické databáze - příprava standby databáze

Kromě primární databáze zahrnuje postup přípravy i kroky nutné provést na stroji určenému pro standby databázi. Prvním z nich je vytvoření LISTENER.ORA, jelikož na

začátku tohoto kroku ještě proces listener a ani soubor LISTENER.ORA na oraguard2 neexistují. Tento soubor lze vytvořit pomocí Netmgr nebo manuálně napsat např. využitím linuxového příkazu `echo` pro zapisování textu do souborů. Konečnou podobu LISTENER.ORA ukazuje obrázek 23. Kromě LISTENER.ORA je opět potřeba připravit TNSNAMES.ORA. Tento soubor narozdíl od prvního souboru je potřeba na obou strojích nakonfigurovat stejně. Tento soubor tedy stačí zkopírovat např. pomocí nástroje scp z lokace primární databáze na stroj s připravovanou standby databází do stejného umístění jako se nachází na prvním počítači [2,13].

Další postup zahrnuje vytvoření souboru s parametry databáze (parameter file). Prozatím stačí, když obsahuje jen jeden parametr a to `DB_NAME`. Tento soubor pak slouží pro spuštění standby databáze ještě před jejím vytvořením. Aby parameter file instance rozeznala, je nutné ho správně pojmenovat. Název musí obsahovat spojení slova `init` a označení `SID` databáze. Vytvořit ho lze obdobně jako při ručním psaní LISTENER.ORA přes příkaz echo (příkaz 5.42). Dalším souborem, který je potřeba připravit, je orapwdoracledb2 (složení slov *orapwd* a `SID` databáze), uchovávající heslo k účtu SYS. Toto heslo se musí shodovat s heslem SYS uživatele na primární databázi. Tudíž je i možné soubor přímo zkopírovat ze stroje s primární databází na připravovaný počítač. Při využití tohoto způsobu je nutné soubor přejmenovat a pro název souboru použít `SID` standby databáze. Druhou možností je využít program orapwd, který je součástí Oracle softwaru a který slouží k vytváření hesla uživatele SYS. Příkazem 5.43 dojde k vytvoření souboru s heslem v požadovaném umístění. Pokaždé, když se změní heslo uživatele SYS primární databáze, tak je vyžadováno nastavit stejná hesla ve všech fyzických standby databázích [2,13].

```
oracle@oraguard1:/opt/oracle/product/12.1.0/
dbhome_1/dbs$
echo db_name= oracle_d > initoracledb2                    (5.42)
oracle@oraguard1:/opt/oracle/product/12.1.0/
dbhome_1/dbs$
./orapwd file=$oracle_home/dbs/orapworacledb2
password=oracle123                                       (5.43)
```

V dalším postupu je potřeba vytvořit adresářovou strukturu pro standby databázi. Jedná se o složku `oradata`, která slouží pro datové soubory databáze, o složku `fast_recovery_area`, do které se ukládají zálohy a soubory využívající recovery operace, a třetí složkou je `adump` pro kontrolní soubory (příkazy 5.44, 5.45, 5.46) [13].

```
oracle@oraguarrd1:~# mkdir -p
/opt/oracle/admin/oracle_db2/adump                    (5.44)
oracle@oraguarrd1:~# mkdir -p /opt/oracle/oradata     (5.45)
oracle@oraguarrd1:~# mkdir -p
/opt/oracle/fast_recovery_area                        (5.46)
```

Na závěr se pomocí SQL*Plus spustí databázová instance pro připravovanou standby (příkaz 5.47). Před jejím spuštěním je potřeba nastavit `ORACLE_HOME` a `ORACLE_SID` jako v případě primární databáze. Databázovou instanci není možné rozbalnout (v režimu `MOUNT`), dokud instance nenalezne řídící soubor (control file). Instanci je tedy nutné spustit v režimu `UNMOUNT`. Tento režim neumožňuje přístup k databázi (v tomto kroku vytváření standby databáze zatím stejně neexistuje), pouze při něm dochází ke spuštění procesů na pozadí a k alokaci paměti [13].

```
SQL> startup nomount pfile=initoracledb2.ora;         (5.47)
```

5.4.3 Vytvoření standby fyzické databáze - proces vytvoření

Nyní přichází na řadu vytvoření standby databáze. K tomuto účelu je určen RMAN, který je součástí Enterprise edice a nachází se v adresáři `/bin` v domovském adresáři Oracle softwaru. Příkaz 5.48 ukazuje spuštění Recovery managera na stroji oraguard1 a příkazy 5.49 a 5.50 následné připojení k oběma databázím.

Po úspěšném připojení lze přejít k finálnímu kroku, kterým je duplikace primární databáze. Proces probíhá při spuštěné databázi, a tedy není nutné žádné přerušení jejího provozu. Vytvoření standby databáze na stroji oraguard2 bylo provedeno pomocí skupiny příkazů 5.51. Příkazy jsou seskupeny do bloku `run`, který postupně vykonává příkazy. RMAN z příkazů v bloku `run` sestaví seznam úkolů a neprodleně je začne vykonávat. První tři řádky `run` bloku umožňují Recovery

manageru využít k duplikaci databáze více procesů současně namísto jednoho, jak je nastaveno ve výchozím nastavení Recovery Manageru, a tím zvýšit rychlost celého procesu duplikace. Další a nejdůležitější příkaz `duplicate` spouští samotnou duplikaci [8].

Poslední sérií příkazů v bloku `run` tvoří parametry standby databáze, které se uloží do souboru server parameter file (spfile). Tento soubor bude později využit k zapnutí databáze namísto provizorního pfile, jenž byl vytvořen při přípravě standby databáze. V spfile dojde k nastavení unikátního jména databáze, umístění řídícího souboru, počtu archivovacích procesů (ARCH procesů), přenosu redo záznamů a prohození názvů `oracle_db1` a `oracle_db2` u datových souborů a u logů. Dále pak nastavení parametrů `fal_server` a `fal_client`. Tyto parametry jsou využity při ztrátě redo záznamu během aktualizace standby databáze tím, že určují, který server má standby databáze požádat o poslání nového redo záznamu [13,14].

```
oracle@oraguard1:/opt/oracle/product/12.1.0/          (5.48)
dbhome_1/bin$ ./rman
```

```
RMAN> connect target sys/oracle123                    (5.49)
RMAN> connect auxiliary sys/oracle123@oracle_db2      (5.50)
```

```
RMAN> run{
allocate channel prmy1 type disk;
allocate channel prmy2 type disk;
allocate auxiliary channel stby type disk;
duplicate target database for standby from active
database
Spfile
parameter_value_convert 'oracle_db1','oracle_db2'
set db_unique_name='oracle_db2'
Set
db_file_name_convert='/oracle_db1/','/oracle_db2/'
set
log_file_name_convert='/oracle_db1/','/oracle_db2/
```

```
'
set
control_files='/opt/oradata/oracle_db2/oracledb2.c
tl'
set log_archive_max_processes='5'
set fal_client='oracle_db2'
set fal_server='oracle_db1'
set standby_file_management='auto'
set
log_archive_config='dg_config=(oracle_db1,oracle_d
b2)'
set log_archive_dest_2='service=oracle_db1 async
valid_for= (online_logfile,primary_role)
db_unique_name=oracle_db1'
;
}
```
(5.51)

Spuštění Redo Apply se provede příkazem 5.52 v SQL*Plus na standby databázi. `using current logfile` znamená, že aktualizace započne ihned po přijetí redo logu. `disconnect` určuje běh aktualizace Redo Apply na pozadí [14].

```
SQL> alter database recover managed standby
database
using current logfile disconnect;
```
(5.52)

V tento moment je docíleno funkční konfigurace i bez využití ODG Brokeru. Přesto další podkapitola popisuje ODG konfiguraci vytvořenou pomocí ODG Brokeru, protože její vytvoření je nutné pro funkční automatický failover.

5.5 Vytvoření a ověření funkčnosti ODG konfigurace

V případě, že standby databáze je korektně vytvořena a nastavena, tak ke zprovoznění základní ODG konfigurace, která má jen jednu standby databázi a nastaveni je ponecháno ve výchozí konfiguraci, stačí pouze několik kroků. Následující text popisuje

vytvoření ODG konfigurace pomocí příkazového řádku DGMGRL, který usnadňuje celý postup vytvoření a spravování ODG konfigurace ve srovnání s použitím SQL*Plus. ODG Broker a jeho příkazovou řádku DGMGRL je potřeba nejdříve zapnout pomocí SQL*Plus (viz příkaz 5.53) [3].

```
SQL> alter system set dg_broker_start = true;          (5.53)
```

Nyní lze příkazový řádek ODG Brokeru spustit. DGMGRL je přístupný v umístění databázového softwaru z adresáře `/bin`. Pro spuštění příkazové řádky je potřeba se přihlásit jako uživatel sys. Na stroji využívající linuxovou distribuci je možné tyto dva kroky vykonat jedním příkazem. Na testovacích strojích byl pro přihlášení využíván příkaz 5.54 [3].

```
oracle@oraguard1:/opt/oracle/product/12.1.0/dbhome_1
/bin$ ./dgmgrl  sys/oracle123                          (5.54)
```

Po úspěšném přihlášení již lze vytvořit ODG konfiguraci. ODG konfigurací může být vytvořeno několik. Proto je nutné jako první vytvoření ODG konfigurace zvolit její název, pod kterým ji lze pak dále spravovat. První část příkazu (příkaz 5.55) vyžaduje jméno ODG konfigurace a ve druhé části je potřeba určit unikátní jméno primární databáze (`DB_UNIQUE_NAME`) [3].

```
DGMGRL> create configuration moje_odg_konfigurace as
primary database is oracle_db1;                        (5.55)
```

Po vytvoření ODG konfigurace a určení primární databáze je dalším nutným krokem přidání alespoň jedné standby databáze, aby ODG konfigurace byla zprovozněna. V příkazu je potřeba opět zadat unikátní jméno databáze, která bude plnit úlohy standby databáze (příkaz 5.56). Posledním krokem pro vytvoření základní ODG konfigurace je její zapnutí (příkaz 5.57) [3].

```
DGMGRL> add database oracle_db2                        (5.56)
DGMGRL> enable configuration;                          (5.57)
```

Správnost ODG konfigurace je zapotřebí ověřit. Jednou z možností je pomocí příkazu 5.58, který zobrazí základní parametry ODG konfigurace. Tento příkaz ukáže zvolenou primární a standby databázi a ochranný režim, jenž je ve výchozím stavu nastaven na maximální výkon. Pro detailnější výpis stačí přidat do příkazu volbu `verbose` (příkaz 5.59). Nejdůležitější informace je na konci výpisu, kde je zobrazen stav konfigurace. V případě správně nastavené a fungující ODG konfigurace je její stav označen jako `SUCCESS`. V opačném případě jsou na konci výpisu zobrazeny chyby způsobující nefunkčnost konfigurace. V případě výskytu chyby není vždy nutné ihned začít hledat řešení, neboť synchronizace se občas může zdržet, a tím způsobit dočasné chyby, které jsou však součástí fungování ODG. Po dokončení synchronizace databází tyto chyby zmizí a konfigurace se přepne do stavu `SUCCESS` (obrázek 25 v příloze zobrazuje funkční konfiguraci) [22].

Vykonáním příkazu 5.60 dojde k zobrazení režimu přenosu mezi primární a standby databází. V tomto kroku je nastaven na asynchronní přenos. Při zapnutém asynchronním přenosu není možné změnit ochranný režim na maximální dostupnost. Nejdříve je nutné pomocí příkazu 5.61 zapnout synchronní přenos a poté již lze pomocí příkazu 5.62 spustit režim maximální dostupnost [43].

```
DGMGRL> show configuration;                                    (5.58)
DGMGRL> show configuration verbose;                            (5.59)
DGMGRL> show database oracle_db2 LogXptMode;                   (5.60)
DGMGRL> edit database oracle_db2 set property
logxptmode='sync';                                             (5.61)
DGMGRL> edit configuration set protection mode as
maxavailability;                                               (5.62)
```

6 Test standby databáze

6.1 Testovací data

Pro otestování standby databáze byla vytvořena tabulka `HODNOCENI_STUDENTU`. Tabulka slouží pro známky studentů, jímž napodobuje ukládání známek na IS/STAG. Tabulka 4 ukazuje použité atributy testovací tabulky. Atribut ID_HODNOCENI slouží pro primární klíč a atribut `DATUM` zaznamenává čas uložení známky do databáze. Zbytek atributů je určen pro vyplnění uživatelem.

ID_HODNOCENI	NUMBER(3,0)
CISLO_STUDENTA	VARCHAR2(20 BYTE)
ZKOUSEJICI	VARCHAR2(30 BYTE)
POKUS	NUMBER(1,0)
DATUM	TIMESTAMP (6)
ZNAMKA	NUMBER(1,0)

Tabulka 4: atributy testovací tabulky - zdroj: vlastní zpracování

Testovací tabulka byla vytvořena pomocí následujícího příkazu 6.1 .

```
CREATE TABLE "ZKOUSEJICI"."HODNOCENI_STUDENTU" (
     "ID_HODNOCENI" NUMBER(3,0) NOT NULL ENABLE,
     "CISLO_STUDENTA" VARCHAR2(20 BYTE),
     "ZKOUSEJICI" VARCHAR2(30 BYTE),
     "POKUS" NUMBER(1,0),
     "DATUM" TIMESTAMP (6),
     "ZNAMKA" NUMBER(1,0),
      CONSTRAINT "ID_H" PRIMARY KEY ("ID_HODNOCENI")
 )                                                          6.1
```

Pro simulování přístupů uživatelů, kteří zapisují známky ze zkoušek byl vytvořen skript (příkaz 6.2). Skript po 10 vteřinách vkládá do vytvořené tabulky známky, čísla studentů a další související hodnoty. Hodnoty známek a čísla pokusů jsou náhodně generovány, do sloupce `ZKOUSEJICI` a `CISLO_STUDENTA` jsou vkládány hodnoty, které obecně označují zkoušející a studenty.

```
declare
i number(2);
a number(2):=1;
begin for i in 1 .. 10 loop

insert into hodnoceni_studentu
(id_hodnoceni, cislo_studenta,zkousejici, pokus,
DATUM, znamka)
values (a, 'student'||a,'zkousejici'||a,(select
trunc(dbms_random.value(1,3),0) from
dual),CURRENT_TIMESTAMP,(select
trunc(dbms_random.value(1,4),0) from dual));
a:=a+1;
insert into hodnoceni_studentu
(id_hodnoceni, cislo_studenta, zkousejici,pokus,
DATUM, znamka)
values (a, 'student'||a,'zkousejici'||a,(select
trunc(dbms_random.value(1,3),0) from
dual),CURRENT_TIMESTAMP,(select trunc
(dbms_random.value(1,4),0) from dual));
a:=a+1;
insert into hodnoceni_studentu
(id_hodnoceni, cislo_studenta, zkousejici, pokus,
DATUM, znamka)
values (a, 'student'||a,'zkousejici'||a,(select
trunc(dbms_random.value(1,3),0) from
dual),CURRENT_TIMESTAMP,(select
trunc(dbms_random.value(1,4),0) from dual));
a:=a+1;

dbms_output.put_line('vkladani znamek'  ||i);
dbms_output.put_line(CURRENT_TIMESTAMP);
commit;
dbms_lock.sleep(10);
end loop;
end;
```

6.2

6.2 Test standby databáze

Průběh testu standby databáze je zachycen pomocí screenshotů terminálu programu Putty. Některé screenshoty jsou oříznané, aby zobrazily pouze důležitou část terminálu. Plnou podobu screenshotů lze nalézt v příloze této práce. První screenshot (obrázek 8) ukazuje vkládání známek do primární databáze pomocí skriptu. Další screenshot (obrázek 9) zachycuje standby databázi. Zde je vidět, že data se kopírují do standby databáze okamžitě, kde jsou k dispozici pro dotazování. Řádky zvýrazněné červenou

barvou ukazují, že skript vložil data do primární databázi v čase 17:44:46 a v čase 17:44:49 už jsou ve standby databázi. Druhý čas neurčuje, kdy byla data zapsána do standby databáze ale říká, kdy byl vykonán dotaz na tyto data. Rychlost kopírování takto malého množství dat z primární do standby databáze trvá méně než vteřinu.

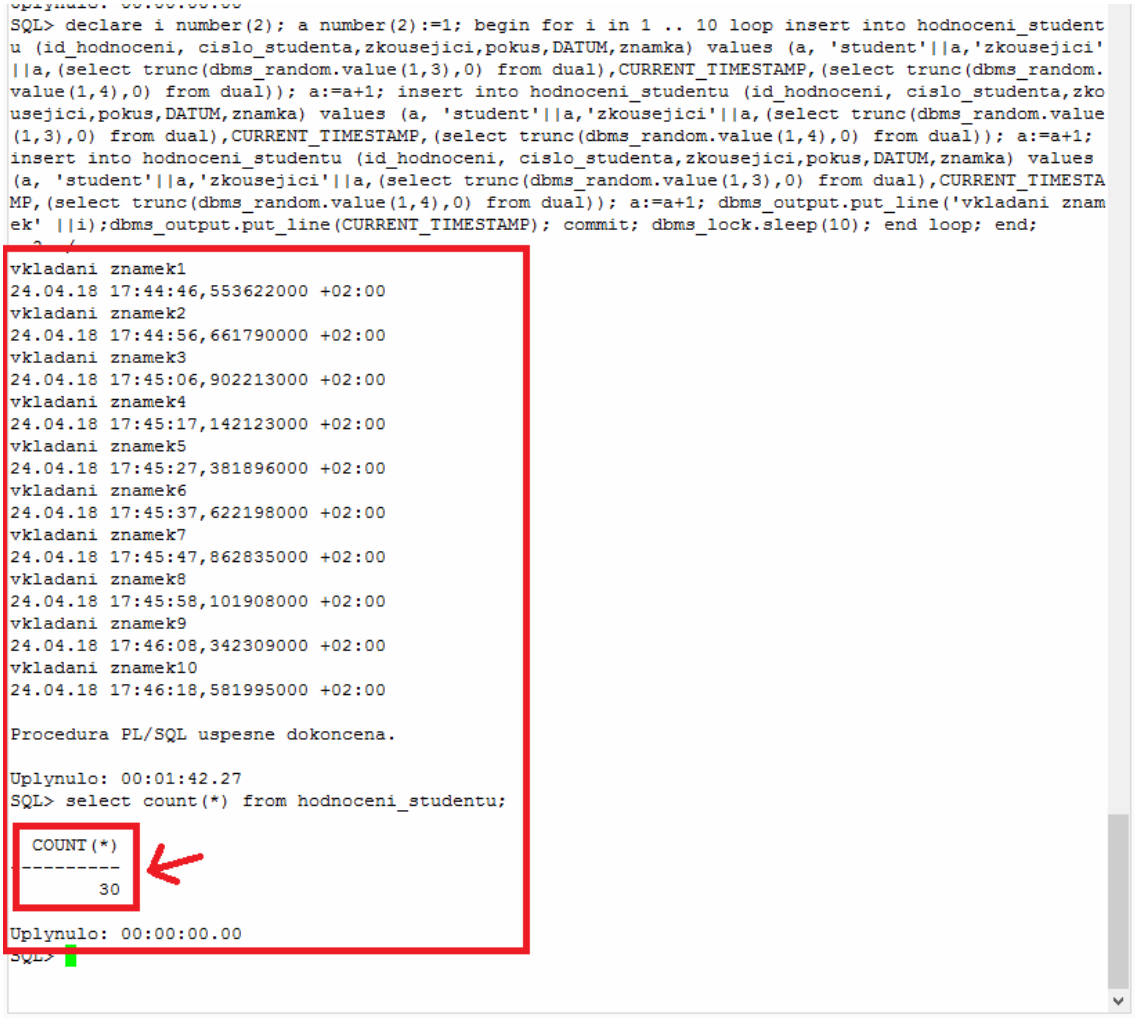
```
SQL> declare i number(2); a number(2):=1; begin for i in 1 .. 10 loop insert into hodnoceni_student
u (id_hodnoceni, cislo_studenta,zkousejici,pokus,DATUM,znamka) values (a, 'student'||a,'zkousejici'
||a,(select trunc(dbms_random.value(1,3),0) from dual),CURRENT_TIMESTAMP,(select trunc(dbms_random.
value(1,4),0) from dual)); a:=a+1; insert into hodnoceni_studentu (id_hodnoceni, cislo_studenta,zko
usejici,pokus,DATUM,znamka) values (a, 'student'||a,'zkousejici'||a,(select trunc(dbms_random.value
(1,3),0) from dual),CURRENT_TIMESTAMP,(select trunc(dbms_random.value(1,4),0) from dual)); a:=a+1;
insert into hodnoceni_studentu (id_hodnoceni, cislo_studenta,zkousejici,pokus,DATUM,znamka) values
(a, 'student'||a,'zkousejici'||a,(select trunc(dbms_random.value(1,3),0) from dual),CURRENT_TIMESTA
MP,(select trunc(dbms_random.value(1,4),0) from dual)); a:=a+1; dbms_output.put_line('vkladani znam
ek' ||i);dbms_output.put_line(CURRENT_TIMESTAMP); commit; dbms_lock.sleep(10); end loop; end;
  2  /
```

Obrázek 8: spuštění skriptu vkládajícího data do primární databáze - zdroj: vlastní zpracování

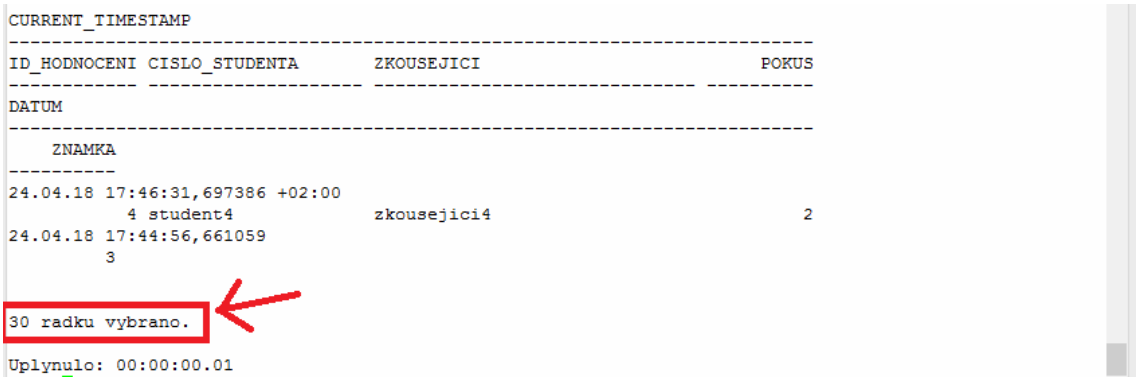
```
oracle@oraguard2: /opt/oracle/product/12.1.0/dbhome_1/bin
30 radku vybrano.

Uplynulo: 00:00:00.01
SQL> select current_timestamp, id_hodnoceni,cislo_studenta,zkousejici,pokus,datum,znamka from hodnocen
i_studentu, dual;

nebyly vybrany zadne radky

SQL> select current_timestamp, id_hodnoceni,cislo_studenta,zkousejici,pokus,datum,znamka from hodnocen
i_studentu, dual;

CURRENT_TIMESTAMP
---------------------------------------------------------------------------
ID_HODNOCENI CISLO_STUDENTA       ZKOUSEJICI                      POKUS
------------ -------------------- ------------------------------ ----------
DATUM
---------------------------------------------------------------------------
    ZNAMKA
----------
24.04.18 17:44:49,881474 +02:00
           1 student1             zkousejici1                           2
24.04.18 17:44:46,552905
         2


CURRENT_TIMESTAMP
---------------------------------------------------------------------------
ID_HODNOCENI CISLO_STUDENTA       ZKOUSEJICI                      POKUS
------------ -------------------- ------------------------------ ----------
DATUM
---------------------------------------------------------------------------
    ZNAMKA
----------
24.04.18 17:44:49,881474 +02:00
           2 student2             zkousejici2                           1
24.04.18 17:44:46,553276
         1


CURRENT_TIMESTAMP
---------------------------------------------------------------------------
ID_HODNOCENI CISLO_STUDENTA       ZKOUSEJICI                      POKUS
------------ -------------------- ------------------------------ ----------
DATUM
---------------------------------------------------------------------------
    ZNAMKA
----------
24.04.18 17:44:49,881474 +02:00
           3 student3             zkousejici3                           1
24.04.18 17:44:46,553434
         2


Uplynulo: 00:00:00.12
```

Obrázek 9: dotaz na data ve standby databázi (stroj: oraguard2) - zdroj: vlastní zpracování

Následují obrázek 10 ukazuje dokonļení skriptu a ļasy jednotlivých ukládání dat do databáze. Do primární databáze skript uloģil 30 záznamŢ. Obrázek 11 zobrazuje pŠístup k datŢm na standby databázi. Na obrázku je vidĐt, ģe standby databáze obsahuje vģech 30 záznamŢ stejnĐjako primární databáze.



Obrázek 10: dokonļení vkládání známek do primární databáze - zdroj: vlastní zpracování



Obrázek 11: standby databáze - dotaz na data - zdroj: vlastní zpracování

6.3 Test Fast-start failover

6.3.1 Příprava Fast-start failover

Popis, co je a k čemu slouží automatický failover (Fast-start failover), je ve třetí kapitole. Tento odstavec popisuje postup nastavení a praktického vykonání automatického failoveru.

Ve výchozím nastavení je automatický failover vypnutý. Ovšem než-li je ho možné spustit, je potřeba inicializovat proces Observer. Tento proces může být v ODG konfiguraci jenom jednou, a proto ho nelze spustit na všech strojích s databázemi. Ideálním řešením je, když proces Observer běží na jiném serveru než běží databáze, aby proces mohl nezávisle sledovat jejich funkčnost a dostupnost. V situaci, kdy je nasazena pouze jedna standby databáze, nebo když jsou k dispozici pouze dva servery, lze vystačit s vytvořením tohoto monitorovacího procesu na počítači se standby databází. V případě, že by počítač se standby databází byl nedostupný, tak i kdyby Observer zůstal v provozuschopném stavu, tak by stejně automatický failover nebylo možné provést. Vytvořit pozorovací proces na primární databázi nemá smysl, neboť při výpadku primární databáze by s velkou pravděpodobností selhal i Observer, a nemohlo by dojít ke spuštění failoveru. Ale ani volba na standby databázi pro běh zmíněného procesu není nikterak vhodná, neboť při neschopnosti primární databáze spojit se s Observerem, může za jistých okolnosti způsobit její výpadek i při funkční konfiguraci [2].

Zapnutí automatického failoveru se skládá ze čtyř kroků. Za prvé je potřeba zvolit standby databázi, která převezme roli primární databáze. V případě nasazení pouze jedné standby lze tento krok přeskočit, jelikož Observer nemá na výběr další standby databáze, na které by přenesl roli primární databáze. Za druhé je potřeba zapnout Flashback technologii na obou databázích. Tento krok byl vykonán při vytváření standby databáze. Jako předposlední krok je potřeba zapnout proces Observer. Ke spuštění Observeru je zapotřebí se přihlásit do DGMGRL, a poté spustit proces. Vhodné je vytvořit log soubor, který zaznamenává průběh failover. Všechny tyto kroky týkající se vytvoření Observeru lze spojit do jednoho příkazu. (příkaz 6.3) [2, 14, 21].

```
oracle@oraguard1:/opt/oracle/product/12.1.0/dbhome
_1/ bin$./dgmgrl sys/password@oracle_db2 "start
observer"
-logfile $HOME/observer.log                    (6.3)
```

```
oracle@oraguard2:~$ cat observer_standby.log

Observer started
[W000 04/10 22:13:43.20] Observer started.
```

Obrázek 12: log soubor Observeru - zdroj: vlastní zpracování

Obrázek 12 ukazuje výpis log souboru Observeru. Dokud neproběhne failover, obsahuje pouze informaci o svém spuštění.

Posledním krokem je nastavení Fast-start failoveru v ODG konfiguraci. Po předchozím kroku již existuje přihlášení do DGMGRL. Proto stačí zadat příkaz 6.4.

Dále lze upravit dva parametry failoveru. Jednak je možné nastavit dobu, po kterou bude Observer po výpadku primární databáze čekat než spustí failover. Příkazem 6.5 je možné nastavit dobu na 30 sekund. A druhým parametrem je tolerovaná doba zpoždění, kterou má standby databáze za primární databází (příkaz 6.6 nastaví dobu na 30 sekund). Proces Observer bere v úvahu tento parametr pouze v případě, že je aktivní režim maximálního výkonu. Pokud zpoždění aktualizace standby databáze je větší než tato definovaná doba, a tím pádem by spuštěním failoveru došlo ke ztrátě netolerovatelnému množství dat, pak Observer failover nespustí. V takové situaci je pak nutný zásah administrátora, aby provedl manuální failover. Obě možnosti jsou ve výchozím stavu nastaveny na 30 vteřin [3].

```
DGMGRL> enable fast_start failover                (6.4)
DGMGRL> edit configuration set property
faststartfailoverthreshold = 30;                  (6.5)
DGMGRL> edit configuration set property
faststartfailoverlaglimit = 30;                   (6.6)
```

Ověření failoveru lze provést pomocí příkazů `SHOW FAST_START FAILOVER` a `SHOW CONFIGURATION VERBOSE`. První výstup DGMGRL pod textem (obrázek 13) zobrazuje parametry failoveru a aktivní podmínky spuštění

automatického failoveru. Druhý výstup z DGMGRL (obrázek 14) ukazuje ODG konfiguraci, tímto výpisem si lze ověřit, že je funkce Fast-start failoveru zapnuta a ODG konfigurace je v chodu.

```
DGMGRL> show fast_start failover

Fast-Start Failover: ENABLED

  Threshold:                 30 seconds
  Target:                    oracle_db2
  Observer:                  oraguard2.zcu.cz
  Lag Limit:                 30 seconds
  Shutdown Primary:          TRUE
  Auto-reinstate:            TRUE
  Observer Reconnect:        (none)
  Observer Override:         FALSE

Configurable Failover Conditions
  Health Conditions:
  Corrupted Controlfile          YES
  Corrupted Dictionary           YES
  Inaccessible Logfile           NO
  Stuck Archiver                 NO
  Datafile Offline               YES

  Oracle Error Conditions:
    (none)
```

Obrázek 13: výstup z DGMGRL zobrazující parametry Fast-start failover - zdroj: vlastní zpracování

```
DGMGRL> show configuration verbose

Configuration - DGC2

Protection Mode: MaxAvailability
Members:
oracle_db1 - Primary database
oracle_db2 - (*) Physical standby database

(*) Fast-Start Failover target

Properties:
FastStartFailoverThreshold       = '30'
OperationTimeout                 = '30'
TraceLevel                       = 'USER'
FastStartFailoverLagLimit        = '30'
CommunicationTimeout             = '180'
ObserverReconnect                = '0'
FastStartFailoverAutoReinstate   = 'TRUE'
FastStartFailoverPmyShutdown     = 'TRUE'
BystandersFollowRoleChange       = 'ALL'
ObserverOverride                 = 'FALSE'
ExternalDestination1             = ''
ExternalDestination2             = ''
PrimaryLostWriteAction           = 'CONTINUE'

Fast-Start Failover: ENABLED

Threshold:                   30 seconds
Target:                      oracle_db2
Observer:                    oraguard2.zcu.cz
Lag Limit:                   30 seconds
Shutdown Primary:            TRUE
Auto-reinstate:              TRUE
Observer Reconnect:          (none)
Observer Override:           FALSE

Configuration Status:
SUCCESS
```

Obrázek 14: výstup DGMGRL zobrazující ODG konfiguraci - zdroj: vlastní zpracování

6.3.2 Klient failover

V situaci, kdy nastane změna rolí primární a standby databází aťuž z důvodu operace failover nebo switchover, tak nedojde automaticky k přepnutí spojení uživatele na novou primární databázi a uživatel by zůstal připojen k původní (předchozí primární) databázi. Aby nebylo zapotřebí po změně role primární databáze manuálně vytvářet pro

uživatele nové spojení, je zapotřebí provést klient failover (nebo také connection failover). Konkrétně se jedná o funkcionalitu failoveru spadající do kategorie Transparent Application Failover (TAF) [3].

Nejdříve je nutné vytvořit službu, prostřednictvím které se uživatel připojí k databázové instanci. K vytvoření této služby slouží PL/SQL balíček `dbms_service`, který definuje databázové služby. Příkaz 6.7 ukazuje proceduru, pomocí níž byla na virtuálním stroji vytvořena služba `oracle_service`. `FAILOVER_METHOD` s hodnotou `BASIC` určuje, že se vytvoří spojení k nové primární databázi. `FAILOVER_TYPE` s hodnotou `SELECT` znamená, že dotazy na data vykonané během operace failover, popř. switchover, (které již nelze vykonat na primární databázi) budou opětovně provedeny na nové primární databázi. `FAILOVER_RETRIES` určuje počet pokusů o připojení na novou primární databázi a `FAILOVER_DELAY` stanovuje dobu mezi jednotlivými pokusy [3].

```
BEGIN
DBMS_SERVICE.CREATE_SERVICE(
service_name=> 'oracle_service',
NETWORK_NAME=> 'oracle_service',
FAILOVER_METHOD=> 'BASIC',
FAILOVER_TYPE=> 'SELECT',
FAILOVER_RETRIES=> 120,
FAILOVER_DELAY=> 1);
DBMS_SERVICE.START_SERVICE(SERVICE_NAME=>
'oracle_service');
end;
/
```

6.7

Ke službě je potřeba vytvořit TRIGGER (6.8), který reaguje na spuštění databáze. TRIGGER zajišťuje, že služba je spuštěna pouze pro specifickou roli [3].

```
CREATE TRIGGER zmena_roli_start
AFTER STARTUP ON DATABASE
DECLARE
V_ROLE VARCHAR2(30);
BEGIN
SELECT DATABASE_ROLE INTO V_ROLE FROM V$DATABASE;
IF V_ROLE = 'PRIMARY' THEN
DBMS_SERVICE.START_SERVICE ('oracle_service');
END IF;
END;
/
```

6.8

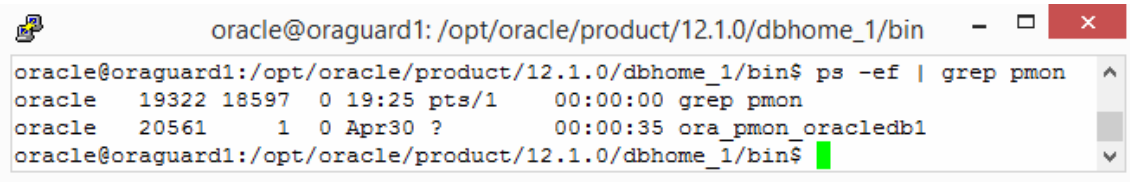
Posledním krokem je úprava TNSNAMES.ORA. Do tohoto souboru je potřeba přidat vytvořenou službu. V TNSNAMES.ORA musí mít služba definované svoje jméno, adresy primární a standby databází a parametr `FAILOVER` s hodnotou `ON` [3]. Konečnou podobu TNSNAMES.ORA lze nalézt v příloze této práce.

6.3.3 Testy funkcí failover a switchover

Průběh testování je zaznamenán pomocí screenshotů. K zachycení času byla využita webová aplikace stopky [46], která je na screenshotech v horních částech. Při vytváření screenshotů bylo potřeba mezi okny přepínat a odmlčovat jednotlivé kroky, což mělo za následek zhoršení přesnosti měření. Na screenshotech se nacházejí především výpisy z příkazové řádky Oracle Data Guard Brokeru. Pokud je ODG Broker zaměstnán probíhající operací failover, tak aktualizuje informace poskytované ve výpisech (jenž jsou obsahem screenshotů) nikdy až po 60 vteřinách. V extrémních případech tedy mohlo při testech dojít ke zhoršení přesnosti měření o 60 sekund. Rozsáhlost schreenshotů neumožňuje jejich vložení přímo do textu této práce, a proto se nacházejí v příloze. Důležité části screenshotů jsou vždy orámované červenou barvou. Následující odstavce se věnují průběhu konkrétního testu automatického failoveru.

Když je Fast-start failover připraven, je potřeba pro automatický failover vyvolat nějakou poruchu. Fast-start failover lze spustit několika způsoby jež byly zmíněny ve třetí kapitole. Nejjednodušším způsobem vyzkoušení failoveru je zastavit databázovou instanci. Pro vyzkoušení na virtuálních strojích byl využit jiný způsob, a to pomocí ukončení procesu PMON. Tento proces se automaticky spouští s každým startem databázové instance. Proces se stará o čištění vyrovnávací paměti databáze a uvolňuje zdroje, které využívají uživatelské procesy (např. vymazává id procesů ze seznamu aktivních procesů). Proces každé tři vteřiny kontroluje, jestli je potřeba provést úklid v paměti. Jeho přítomnost je nutná pro běžící databázi, a v případě jeho ukončení dojde k uzavření databáze.

K ukončení procesu je potřeba zjistit jeho process identification (PID). K využitým příkazům není potřeba oprávnění root a lze je tedy spustit přes vytvořeného uživatele `oracle`. Ke zjištění slouží kombinace linuxových nástrojů ps a grep, které umožňují vybrat konkrétně proces PMON (obrázek 15).



Obrázek 15: proces PMON na stroj oraguard1 - zdroj: vlastní zpracování

Příkaz ps vypsal dva řádky s procesem PMON (obrázek 15). Hledaný proces PMON obsahuje ve svém názvu `SID` databáze, kterou sleduje. Primární databáze `oracle_db1` nese `SID oracledb1`. Podle obrázku je PID hledaného procesu 1080. Dalším krokem je ukončení procesu PMON příkazem `kill -9 1080`, což následně vyvolá výpadek databáze. Volba `9` v příkazu způsobí, že k ukončení dojde neprodleně. Následující číslování v textu (1.-10.) shrnuje kroky testu do jednotlivých milníků, které odpovídají obrázkům v příloze.

1. V čase 0 minut 0 sekund selhala primární databáze vypnutím procesu PMON

Po vypnutí procesu PMON dojde k uzavření databáze. Obrázek 31 ukazuje chybu ORA-01034, která oznamuje, že primární databáze je nedostupná. Zjištění, že primární databáze je nedostupná trvalo necelých deset vteřin. Po tomto zjištění Observer začal odpočítávat nastavený čas 30 sekund než spustí failover. Po 30 vteřinách již ODG Broker ukazuje (obrázek 32), že je spuštěn failover.

2. V čase 0 minut 10 sekund je spuštěn odpočet Observeru (obrázek 31).
3. V čase 0 minut 42 sekund je zahájen failover (obrázek 32).

Po 36 vteřinách je failover dokončen. Nová primární databáze je `oracle_db2` a databáze `oracle_db1` je offline (obrázek 33). V této situaci ODG konfigurace hlásí chyby, protože databáze nedokáží provést synchronizaci, a je potřeba bývalou primární databázi nejdříve opravit. K opravení obou chyb dojde po obnovení databáze `oracle_db1`.

4. V čase 1 minuta 8 sekund je připravena nová primární databáze (obrázek 33).

Ke spuštění databáze `oracle_db1` (v případě selhání procesu PMON) stačí vykonat příkaz `startup` v SQL*Plus (obrázek 34). Databáze `oracle_db1` se po spuštění automaticky začne převádět na standby databázi a poté se provede synchronizace s primární databází (obrázek 35, obrázek 36). Po 63 sekundách, od spuštění současné standby databáze, je ODG konfigurace plně funkční (obrázek 37). Celý proces failoveru i s opětovným zprovozněním ODG konfigurace trval 180 vteřin. Nyní existuje funkční ODG konfigurace, ale s prohozenými rolemi. Primární databáze je `oracle_db2` a standby databáze je `oracle_db1`. Pro navrácení se na původní ODG konfiguraci lze využít operaci switchover.

5. V čase 1 minuta 56 sekund je obnovena databáze po výpadku (obrázek 34).
6. V čase 2 minuty 26 sekund probíhá příprava nové standby databáze (obrázek 35).
7. V čase 2 minuty 42 sekund probíhá synchronizace primární a standby databází (obrázek 36).
8. V čase 3 minuty je ODG konfigurace plně provozuschopná (obrázek 37).

Operaci switchover lze provést pomocí DGMGRL na jakékoli funkční ODG konfiguraci. příkazem 6.9 se spustí přepnutí rolí (obrázek 38). Po dokončení procesu je potřeba zkontrolovat konfiguraci např. příkazem 6.10. Obrázek 38 ukazuje nastalou chybu ORA-16525, která signalizuje, že proces ODG Brokeru selhal. Tato chyba sice nesignalizuje nejlepší průběh switchoveru, ale představuje pouze dočasné zdržení. Tato chyba je jednou z mnoha, které se mohou objevit při synchronizaci databází prováděné po delší době nebo po výpadku databáze. řádově po desítkách vteřin tyto chyby zmizí. Samotný switchover trval 55 sekund. Obnovení ODG konfigurace do plně funkčního stavu dalších 54 vteřin (obrázek 39, obrázek 40). Celkový čas od selhání primární databáze s vykonáním failoveru a switchoveru až po navrácení se na původní ODG konfiguraci trval 5 minut a 35 vteřin.

```
DGMGRL> switchover to oracle_db1                    (6.9)
DGMGRL> show configuration                          (6.10)
```

9. V čase 4 minuty 41 sekund je dokončena operace switchover (obrázek 38).

10. V ļase 5 minut 35 sekund je ODG konfigurace navrácena do pŢvodního stavu a zcela funkļní (obrázek 39 zobrazuje oraguard1, obrázek 40 ukazuje oraguard2).

KromŀDGMGRL lze informace získat z logu Observeru. Obsah logu po provedení failoveru je pod textem (obrázek 16). Oproti pŚedchozímu zobrazení logu pŚibyl záznam o failoveru. Log ukazuje, kdy byl failover proveden a mezi jakými databázemi. Podle záznamu zprovoznŀní konfigurace trvalo 120 sekund. Rozdíl oproti namŀŚeným 180 sekundám lze vysvŀtlit jednak postupem mŀŚení (jenģ byl popsán na zaļátku kapitoly 6.3.3.) a jednak tím, ģe Observer poļítá pouze dobu, za kterou jsou obŀ databáze opŀt v provozu, a mohou spolu komunikovat, a uģ nezapoļítává dalģí ļas, potŚebný na synchronizaci databází, a který je v mŀŚených 180 sekundách zapoļítán.

```
oracle@oraguard2:~$ cat observer_standby.log

Observer started
[W000 04/16 11:42:20.14] Observer started.

14:25:11.16  Monday, April 16, 2018
Initiating Fast-Start Failover to database "oracle_db2"...
Performing failover NOW, please wait...
Failover succeeded, new primary is "oracle_db2"
14:25:15.28  Monday, April 16, 2018

14:26:52.16  Monday, April 16, 2018
Initiating reinstatement for database "oracle_db1"...
Reinstating database "oracle_db1", please wait...
Reinstatement of database "oracle_db1" succeeded
14:27:11.29  Monday, April 16, 2018
```

Obrázek 16: log soubor zaznamenávající failover - zdroj: vlastní zpracování

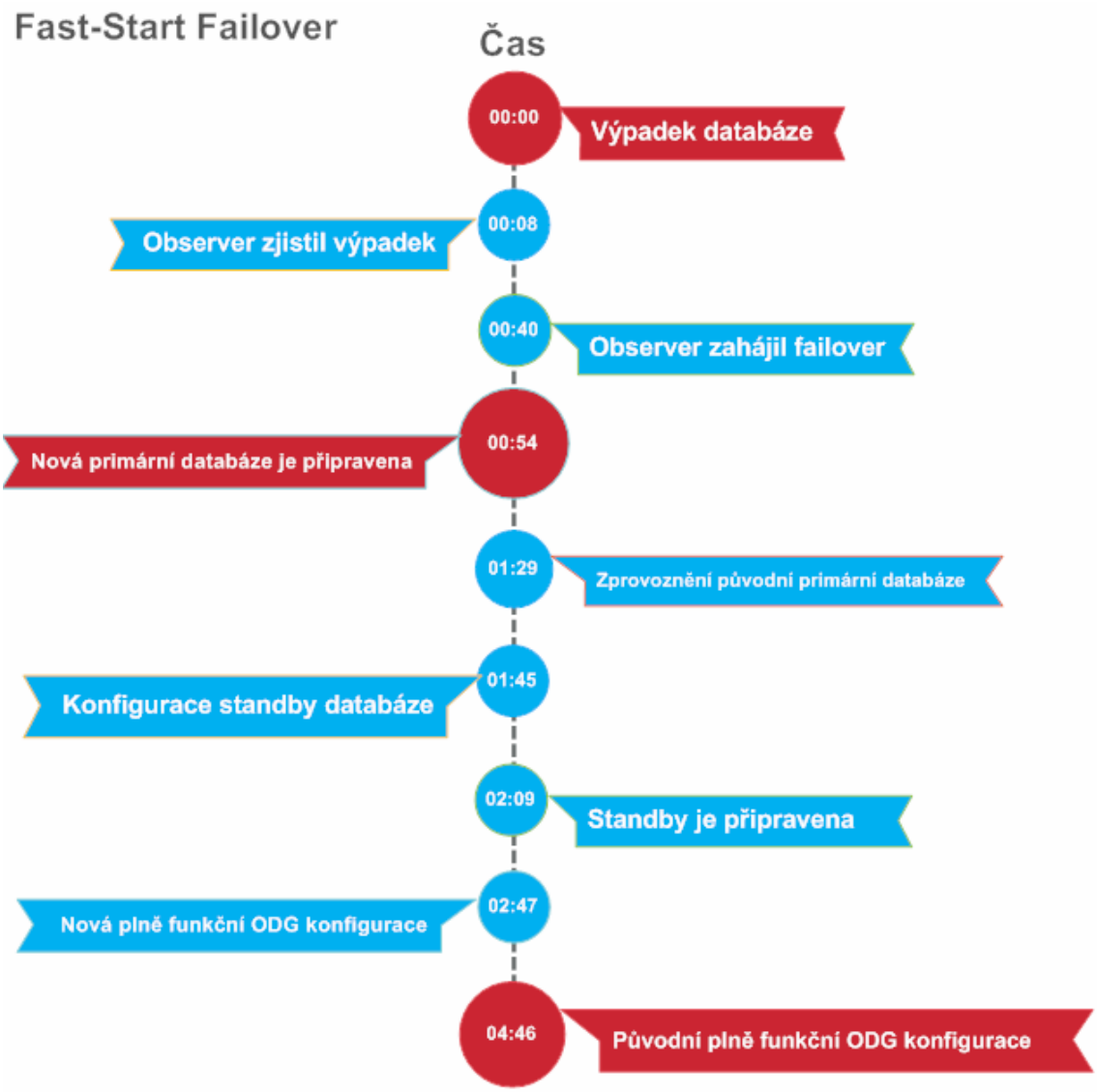
Celkem bylo provedeno 20 testŢ automatického failoveru (tabulka 5). Obrázek 17 zobrazuje prŢmŀrné ļasy jednotlivých krokŢ tŀchto testŢ. Aģ na prŢbŀh switchoveru (krok ļíslo 9, obrázek 38), který je v tŀchto testech slouļen s následujícím krokem (pŚi nŀkterých testech bylo obtíģné tento krok zaznamenat a jeho zmŀŚený ļas by byl velmi nepŚesný), se jednotlivé kroky shodují s pŚedchozím testem.

Číslo operací:

1. Proces Observer zjistil selhání primární databáze
2. Observer spustil failover
3. Nová primární databáze je připravena
4. Manuální obnovení předchozí primární databáze
5. Automatické zahájení přípravy nové standby databáze
6. Probíhající synchronizace mezi databázemi
7. Nová ODG konfigurace je plně funkční
8. Obnovení původní konfigurace (pomocí switchoveru)
9. Doba failoveru z pohledu Observeru (údaj z logu Observeru)

Číslo testu	**Číslo operace**								
	1	2	3	4	5	6	7	8	9
1	7	34	58	113	137	166	218	342	140
2	3	30	51	86	89	89	141	252	94
3	7	38	55	90	114	127	142	272	90
4	7	40	53	93	112	134	141	272	93
5	5	37	47	88	106	122	178	285	87
6	5	39	51	90	106	123	178	340	87
7	8	42	55	93	115	138	148	261	95
8	10	45	56	93	115	138	145	307	98
9	15	50	90	100	127	143	153	266	352
10	7	37	49	87	113	131	136	239	92
11	6	42	57	98	120	159	225	336	123
12	7	40	51	85	111	124	171	277	83
13	7	38	48	83	105	119	179	290	83
14	15	44	54	89	112	134	142	256	85
15	9	39	51	86	106	121	173	286	85
16	5	36	47	80	100	120	171	276	82
17	5	37	47	78	102	121	177	286	79
18	13	48	57	88	112	123	184	344	78
19	5	38	47	81	101	124	173	270	82
20	5	38	48	80	110	123	166	270	81
Průměr	8	40	54	89	105	129	167	286	104

Tabulka 5: výsledky testování failoveru (v sekundách) - zdroj: vlastní zpracování



Obrázek 17: prŢbh failoveru a navrácení do pŢvodní konfigurace - zdroj: vlastní zpracování

6.3.4 Zhodnocení poļtu testŢ

Po skonļení testŢ je potŚeba ovlŃit, zda jejich poļet je dostaļující. K výpoļtŢm jsou vyuģity koneļné ļasy jednotlivých testŢ. OvlŃení je provedeno pomocí statistických ukazatelŢ. Prvním ukazatel je smŃrodatná odchylka, která je charakteristikou variability. Pro poļet mŃŚení, kterých je ménŃ neģ 30, se pouģívá výbŃrová smŃrodatná odchylka (rovnice 1) [45].

$$s^2 = \frac{1}{n-1} * \sum \left(x_i - \bar{x} \right)$$

(rovnice 1)

s - výběrová směrodatná odchylka

n - počet pokusů

x_i - i-té měření

$x_{\leftarrow}$ = průměr měření

Dosazením do vzorce vyjde hodnota výběrové směrodatné odchylky 31,41. Vhledem ke zpoždění některých výpisů až o 60 sekund (viz kapitola 6.3.3), se dala tato hodnota očekávat. Ve vzdálenosti 31,41 od hodnoty průměru se nachází 70 procent hodnot měření, což přibližně odpovídá hodnotě normálního rozdělení (68 procent).
Dalším ukazatelem je přípustná chyba (rovnice 2). Kvantil je možné zjistit z tabulek. Při zvolení hladiny významnosti 5% je hodnota kvantilu 1,96 [45].

$$\varphi = \frac{\hat{\sigma}}{\sqrt{n}} * u_{1-2\alpha} \qquad \text{(rovnice 2)}$$

n - počet testů

φ - přípustná chyba

$u_{1-2\alpha}$ - kvantil normovaného normálního rozdělení

$\hat{\sigma}$ - směrodatná odchylka

α - hladina významnosti

V ideálním případě by byla velikost chyby nulová. K velikosti chyby, která by se blížila k nule by však bylo zapotřebí tisíce testů. Po 20 testech je velikost přípustné chyby 13,77. To znamená, že skutečná průměrná doba zprovoznění konfigurace po výpadku se může lišit od vypočítané o 13,77 sekund. Hodnoty pro posouzení velikosti chyb nejsou pevně dány. Velikost chyby, která je ještě uspokojivá, je dána podstatou testu. Vzhledem k povaze testu a způsobu měření lze tuto velikost chyby posoudit jako přiměřenou a počet testů lze považovat za dostatečný. V příloze se nachází tabulka s mezivýpočty (tabulka 6).

6.4 Přínos z pohledu uživatele

Z pohledu běžného uživatele funkce Fast-start failover přináší nepřerušovaný přístup k datům v databázi. Společnost Oracle uvádí dobu výpadku maximálně v řádku

jednotek vteřin [29]. Na virtuálních strojích z probíhlých měření vyšla průměrná doba automatických failoverů 24 sekund. Celková doba, za kterou je nová primární databáze připravena i se započítanou dobou 30 sekund, po kterou Observer čeká, je 54 sekund (tabulka 5).

Obrázek 18 zachycuje sedm dotazů na data v primární databázi, představující přístupy uživatelů k datům. Dotazy byly spouštěny s jednosekundovou prodlevou. Uživatel na výsledek dotazu musel čekat méně než jednu vteřinu. Mezi třetím a čtvrtým dotazem došlo k výpadku primární databáze (obrázek 18 v červeném rámečku). Obrázek ukazuje, že na požadovaná data musel uživatel čekat 39 sekund. Následující dotazy probíhaly na nové primární databázi obdobnou rychlostí jako dotazy před výpadkem.

Obrázek 18: failover z pohledu uživatele - zdroj: vlastní zpracování

7 Závěr

Cílem této práce je vybrat vhodnou variantu technologie Oracle Data Guard, vybranou možnost implementovat a ukázat přínosy pro informační systém Západočeské univerzity STAG a především pak přínosy z pohledu běžného uživatele informačního systému, jemuž tato technologie je schopna poskytnout téměř nepřetržitou dostupnost dat a umožňuje zrychlení přístupu k datům IS/STAG.

Prvním cílem práce bylo seznámit se s databází Oracle v Enterprise edici ve verzi 12.1.0.2.0 a technologií Oracle Data Guard, která je součástí této edice. Prozkoumat architekturu Oracle databáze a Oracle Data Guard. Vedle architektury byla věnována pozornost funkcím, jež technologie Oracle Data Guard nabízí.

Dále byly prostudovány jednotlivé typy standby databází a na základě zjištěných informací byly analyzovány jejich schopnosti. Pro implementaci bylo potřeba se seznámit i s ochrannými režimy, v nichž standby databáze bude pracovat, a stejně jako u typů standby provést jejich porovnání.

Následně byl vytvořen výčet kladů a záporů jednotlivých standby databází (kapitola 4 - tabulka 1) a ochranných režimů (kapitola 4 - tabulka 2). Vytvořený seznam výhod a nevýhod standby databází a ochranných režimů byl zkonzultován s vedoucím této práce. Na základě toho vyšla pro potřeby IS/STAG nejlépe fyzická standby databáze. Cílem implementovaného řešení byla především ochrana dat produkční databáze, což nejlépe splňuje právě fyzická standby databáze, která jako jediná je schopna chránit veškerá data produkční databáze IS/STAG. Hlavní zápor logické standby je, že nepodporuje datový typ Collection. U snapshot standby je nevýhodou, že neobsahuje aktuální data. Pro standby databázi byl zvolen ochranný režim maximální dostupnost. V režimu maximální ochrany jsou data primární databáze lépe chráněna. Tento režim však nepodporuje funkci automatického failoveru a proto byl zamítnut.

Po konzultaci s vedoucím práce byly pro implementaci vybrány virtuální stroje CIVu, které simulovaly prostředí IS/STAG. Z bezpečnostních důvodů nebyla implementace provedena přímo na produkční databázi IS/STAG. IS/STAG je zásadní systém pro univerzitu a je nutné aby fungoval nepřetržitě a je vyloučené na něm provádět jakékoli testy, jenž by mohly skončit nedostupností dat. Virtuální stroje byly vytvořeny tak, aby napodobovali prostředí produkční databáze IS/STAG a jsou plně dostačující pro ověření funkčnosti technologie Oracle Data Guard a předvedení jejích přínosů pro IS/STAG. Jediným rozdílem může být rychlost prováděných transakcí,

které proběhly v prvním testovacím případu standby databáze při velkém zatížení databáze. Testy při velké zátěži nemohly být provedeny z důvodu nedostatku místa na pevném disku. V reálném nasazení by při plném zatížení IS/STAG mohly transakce trvat oproti výsledku testovacího případu o něco déle. Oracle uvádí [47], že při zatížení 6000-mi transakcemi za vteřinu, (což bohatě přesahuje nároky IS/STAG) a při latenci 1 ms mezi primární a standby databází je snížení počtu vykonaných transakcí za jednu vteřinu o tři procenta. Konkrétní hodnota pro IS/STAG by však záležela na použitém hardwaru pro standby databázi a na spojení s produkční databází.

Součástí práce je popis přípravy prostředí a konfigurace Oracle Data Guard. Tento postup je možné využít i při implementaci na reálné prostředí IS/STAG, který běží na stejném operačním systému jako virtuální stroje. Dále byly provedeny testy simulující provoz IS/STAG. První testování standby databáze bylo provedeno na testovacím případu, který reálně odpovídá situaci, kdy zkoušející vkládá známky vyzkoušených studentů do IS/STAG. Zkoušející byl nahrazen testovacím skriptem, který postupně i s časovými prodlevami (aby testovací skript odpovídal realitě) vkládal jednotlivé známky ze zkoušky do systému. Při vkládání testovacího vzorku nebyla z pohledu uživatele velikost zpoždění vykonání transakce (zpoždění z důvodu duplikace do standby) zaznamenatelná. V případě využití režimu maximální ochrany v reálném nasazení je potřeba zvolit spojení mezi databázemi dostatečnou latencí a šířkou pásma. Alternativou by pak mohl být režim maximálního výkonu, který má zanedbatelný vliv na výkon produkční databáze. Poté byl testován automatický failover, který ukázal chování při výpadku produkční databáze a jehož výsledkem je průměrný čas trvání failoveru. Průběh tohoto testovacího případu je pouze minimálně ovlivněn prostředím [48] a tudíž lze očekávat téměř stejné výsledky i v případě nasazení na IS/STAG. Na závěr bylo ukázáno, jak failover vidí běžný uživatel.

Pouģité zkratky

ARCH	Archiver proces
ASM	Automatic Storage Management
DBMS	Database Management System
DBW	Database Writter
DBWn	n-tý Database Writter
DBWR	Database Writter (starģí oznaļení)
EE	Enterprise edice
CKPT	Checkpoint process
FAL	Fetch Archive Log
FCI	Failover Cluster Instance
HADR	High Availability Disaster Recovery
IS	Informaļní systém
LGWR	Log Writer
LRU	Least Recently Used
LSP	Logical Standby Process
MAA	Maximum Availability Architecture
MRP	Managed Recovery Process
ODG	Oracle Data Guard
PGA	Program Global Area
RAC	Real Application Clusters
RDBMS	Relational Database Management System
RFS	Remote File Server
RMAN	Recovery Manager
SGA	System Global Area
STAG	Studijní Agenda
TAF	Transparent Application Failover
WSFC	Windows Server Failover Clustering

Literatura

[1] JESSE, Scott, Bill BURTON a Bryan VONGRAY. *Oracle database 11g release 2 high availability maximize your availability with grid infrastructure, Oracle real application clusters, and Oracle data guard*. 2nd ed. New York: McGraw-Hill, 2011. ISBN 9780071752077.

[2] LARRY CARPENTER ... [ET AL.]. Oracle *data guard 11g handbook*. New York: Oracle Press/McGraw-Hill, 2009. ISBN 9780071621489.

[3] BARANSEL, Emre a Nassyam BASHA. *Oracle Data Guard 11gR2 administration beginner's guide learn how to build and maintain Data Guard configurations with real-life, practical examples*. Online-Ausg. Birmingham: Packt Pub, 2013. ISBN 9781849687904.

[4] THOMAS, Biju. *OCA: Oracle database 11g administrator certified associate study guide* [online]. Indianapolis, Indiana: Wiley Publishing, 2009 [cit. 2018-04-29]. Serious skills. ISBN 978-047-0395-127. Dostupné z: https://thepankajpanigrahi.files.wordpress.com/2011/09/ebook.pdf

[5] EBBERS, Mike, Kathryn ARRELL, Sam AMSAVELU, Gaylan BRASELTON, Terry ELLIOTT, Leon RICH, Barton ROBINSON a David SIMPSON. *Experiences with Oracle Database 12c Release 1 on Linux on System z* [online]. International Business Machines Corporation, 2014 [cit. 2018-04-25]. ISBN 073843938X. Dostupné z: http://www.redbooks.ibm.com/redbooks/pdfs/sg248159.pdf

[6] KREINES, David C. *Oracle DBA: kapesní průvodce*. Praha: Grada, 2006. ISBN 80-247-1669-0.

[7] KUMAR, Bipul. *Oracle data guard: standby database failover handbook*. [Nachdr.]. Kittrell, NC: Rampant TechPress, 2007. ISBN 0974599387.

[8] KUHN, Darl. *Oracle RMAN database duplication*. New York: Apress, 2015. ISBN 978-1-4842-1113-7.

[9] BAUMANN, Bernhard. Setting up Oracle 12c Data Guard for SAP customers. *SAP* [online]. Walldorf: Oracle Support Solution Center SAP, 2015 [cit. 2018-04-25]. Dostupné z: https://www.sap.com/documents/2016/12/a67bac51-9a7c-0010-82c7-eda71af511fa.html

[10] Data Guard Physical Standby Setup in Oracle Database 11g Release 2. *Oracle-base* [online]. Birmingham, UK: Tim Hall, c2000-2018 [cit. 2018-04-25]. Dostupné z: https://oracle-base.com/articles/11g/data-guard-setup-11gr2#primary_server_setup

[11] Data Guard Physical Standby Setup Using the Data Guard Broker in Oracle Database 12c Release 1. *Oracle-base* [online]. Birmingham, UK: Tim HALL, c2000-2018 [cit. 2018-04-27]. Dostupné z: https://oracle-base.com/articles/12c/data-guard-setup-using-broker-12cr1

[12] SOLAř , Tomáġ. Jak vlastnĐdatabáze Oracle pracuje?. Tomas-solar [online]. Tomáġ SolaŚ, 2013 [cit. 2018-04-25]. Dostupné z: http://www.tomas-solar.com/blog/jak-vlastne-databaze-oracle-pracuje/

[13] KHLIFI, Wissem El. Data Guard Physical Standby Database Best Practices ï Part 1. *Red-gate* [online]. Cambridge: Red Gate Software, 2012, 07 August 2012 [cit. 2018-04-25]. Dostupné z: https://www.red-gate.com/simple-talk/sql/oracle/data-guard-physical-standby-database-best-practices-part-i/

[14] KHLIFI, Wissem El. Data Guard Physical Standby Database Best Practices ï Part II. *Red-gate* [online]. Cambridge: Red Gate Software, 2012, 27 September 2012 [cit. 2018-04-25]. Dostupné z: https://www.red-gate.com/simple-talk/sql/oracle/data-guard-physical-standby-database-best-practices-part-ii/

[15] RUDD, Warwick. SQL Server 2012 AlwaysOn. *Red-gate* [online]. Cambridge: Red Gate Software, 2012, 15 May 2012 [cit. 2018-04-25]. Dostupné z: https://www.red-gate.com/simple-talk/sql/database-administration/sql-server-2012-alwayson/

[16] How to Install Oracle 11gR2 on Debian vserver. *U1research* [online]. Budapest: U1 Research, c2018 [cit. 2018-04-25]. Dostupné z: http://u1research.org/blogs/en/2010/05/17/oracle11gr2-debian-vserver/

[17] JESSE, Scott, Bill BURTON a Bryan VONGRAY. Best Oracle Data Guard Standby Database Configuration?. *Logicalread* [online]. Austin: SolarWinds Worldwide, c2018 [cit. 2018-04-25]. Dostupné z: https://logicalread.com/best-oracle-data-guard-standby-database-configuration-mc04/#.WuDWrH9pFF1

[18] Architecture. *Datadisk* [online]. c2015 [cit. 2018-04-25]. Dostupné z: http://www.datadisk.co.uk/html_docs/oracle_dg/architecture.htm

[19] Configure physical standby database Oracle Database 11G release 2 (11.2). *Dbaora* [online]. Tomasz Lesinski, 2014 [cit. 2018-04-25]. Dostupné z: http://dbaora.com/configure-physical-standby-database-oracle-database-11g-release-2-11-2/

[20] CHMEL, Marek. SQL Server a vysoká dostupnost ï I. *Zive* [online]. CN Invest a.s, 2012 [cit. 2018-04-27]. Dostupné z: https://www.zive.cz/clanky/sql-server-a-vysoka-dostupnost--i/sc-3-a-165359/default.aspx

[21] Fast-Start Failover - It is reliable. *Oracle-tech.blogspot* [online]. Mihajlo Tekic, 2008 [cit. 2018-04-27]. Dostupné z: http://oracle-tech.blogspot.cz/2008/08/fast-start-failover-it-is-reliable.html

[22] My Oracle World. *Oracle-tech.blogspot* [online]. Mihajlo Tekic, 2013 [cit. 2018-04-27]. Dostupné z: http://oracle-tech.blogspot.cz/2013/

[23] Fast-Start Failover in Oracle 11g Data Guard. *Databasejournal* [online]. Foster City: QuinStreet, 2009 [cit. 2018-04-27]. Dostupné z: https://www.databasejournal.com/features/oracle/article.php/3849106/Fast-Start-Failover-in-Oracle-11g-Data-Guard.htm

[24] Steps to configure Oracle 11g Data Guard Physical Standby ï Active Data Guard Part-I. *Dbatricksworld* [online]. Jignesh Jethwa, 2014 [cit. 2018-04-27]. Dostupné z: https://dbatricksworld.com/steps-to-configure-oracle-11g-data-guard-physical-standby-data-guard-part-i/

[25] Create Physical standby database using RMAN -- ORACLE DATAGUARD. *Practical DBA XP - Try on your own!* [online]. London, United Kingdom: Vijendra Kalburgi, 2011 [cit. 2018-04-27]. Dostupné z: http://pracdba.blogspot.cz/2011/08/create-physical-standby-database-using.html

[26] *Dbvisit Software* [online]. Atlassian Corporation, c2003-2017 [cit. 2018-04-27]. Dostupné z: https://dbvisit.atlassian.net/wiki/

[27] RIES, Steve. *OCA Oracle Database 11g: database administration I : a real-world certification guide*. New Edition. Birmingham: Packt Pub, 2013. ISBN 9781849687300.

[28] RITCHIE, Colin. *Relational database principles*. 2nd ed. London [u.a.]: Continuum, 2002. ISBN 0826457134.

[29] Switchover and Failover Operations. *Docs.oracle* [online]. Toronto: Oracle [cit. 2018-05-01]. Dostupné z: https://docs.oracle.com/database/121/DGBKR/sofo.htm#DGBKR390

[30] Introduction to Oracle Data Guard. *Docs.oracle* [online]. Toronto: Oracle [cit. 2018-05-01]. Dostupné z: https://docs.oracle.com/database/121/SBYDB/concepts.htm#SBYDB5252

[31] Oracle Database Editions. *Docs.oracle* [online]. Toronto: Oracle [cit. 2018-05-01]. Dostupné z: https://docs.oracle.com/database/121/DBLIC/editions.htm#DBLIC109

[32] CONNOLLY, Thomas M. a Carolyn E. BEGG. *Database systems: a practical approach to design, implementation, and management*. 4th ed. New York: Addison-Wesley, c2005. ISBN 0321210255.

[33] MIKOLÁĠEK, Jan. *ř eġení vysoké dostupnosti databáze z pohledu neplánovaných výpadkŢ* [online]. Praha, 2014 [cit. 2018-05-04]. Dostupné z: https://www.unicorncollege.cz/bakalarske-prace/archiv-2014/mikolasek-jan/attachments/BP_-_Mikolasek_Jan.pdf. BakaláŚská práce. Unicorn College.

[34] HOLÝ, Ludĺk. *Ochrana dat Oracle databázového stroje* [online]. Praha, 2010 [cit. 2018-05-04]. Dostupné z: https://is.bivs.cz/th/fkma1/bp_Holy_Ludek_uco10598.pdf. Bakaláśská práce. Vysoká ġkola regionálního rozvoje a Bankovní institut.

[35] Company Information. *Oracle* [online]. Toronto: Oracle [cit. 2018-05-04]. Dostupné z: https://www.oracle.com/corporate/index.html

[36] Logical Storage Structures. *Docs.oracle* [online]. Toronto: Oracle [cit. 2018-05-04]. Dostupné z: https://docs.oracle.com/cloud/latest/db112/CNCPT/logical.htm#CNCPT301

[37] Oracle Architecture. *Mahameeditpro.blogspot* [online]. Hyderabad, India: MOHAMMED ABDUL HAMEED, 2016 [cit. 2018-05-04]. Dostupné z: http://mahameeditpro.blogspot.cz/2016/09/oracle-architecture.html

[38] Sql-server-2017. *Microsoft* [online]. Redmond, Washington: Microsoft [cit. 2018-05-04]. Dostupné z: https://www.microsoft.com/en-us/sql-server/sql-server-2017

[39] Real Application Clusters (RAC). *Intellipaat* [online]. intellipaat.com, c2011-2018 [cit. 2018-05-04]. Dostupné z: https://intellipaat.com/tutorial/oracle-dba-tutorial/real-application-clusters-rac/

[40] Far Sync. *Docs.oracle* [online]. Toronto: Oracle [cit. 2018-05-05]. Dostupné z: https://docs.oracle.com/database/121/SBYDB/create_fs.htm#SBYDB5416

[41] Managing Physical and Snapshot Standby Databases. *Docs.oracle* [online]. Toronto: Oracle [cit. 2018-05-05]. Dostupné z: https://docs.oracle.com/database/121/SBYDB/manage_ps.htm#SBYDB00705

[42] Using SQL Apply to Upgrade the Oracle Database. *Docs.oracle* [online]. Toronto: Oracle [cit. 2018-05-05]. Dostupné z: https://docs.oracle.com/cd/E18283_01/server.112/e17022/rollup.htm

[43] Scenarios Using the DGMGRL Command-Line Interface. *Docs.oracle* [online]. Toronto: Oracle [cit. 2018-05-05]. Dostupné z: https://docs.oracle.com/database/121/DGBKR/cli.htm#DGBKR505

[44] Oracle. *Wiki.debian* [online]. 2017 [cit. 2018-05-11]. Dostupné z: https://wiki.debian.org/DataBase/Oracle

[45] ĠPICAR, Radim. *Pr*Ţ*vodce Statistikou*.

[46] *Stopky.info* [online]. [cit. 2018-05-12]. Dostupné z: https://stopky.info/

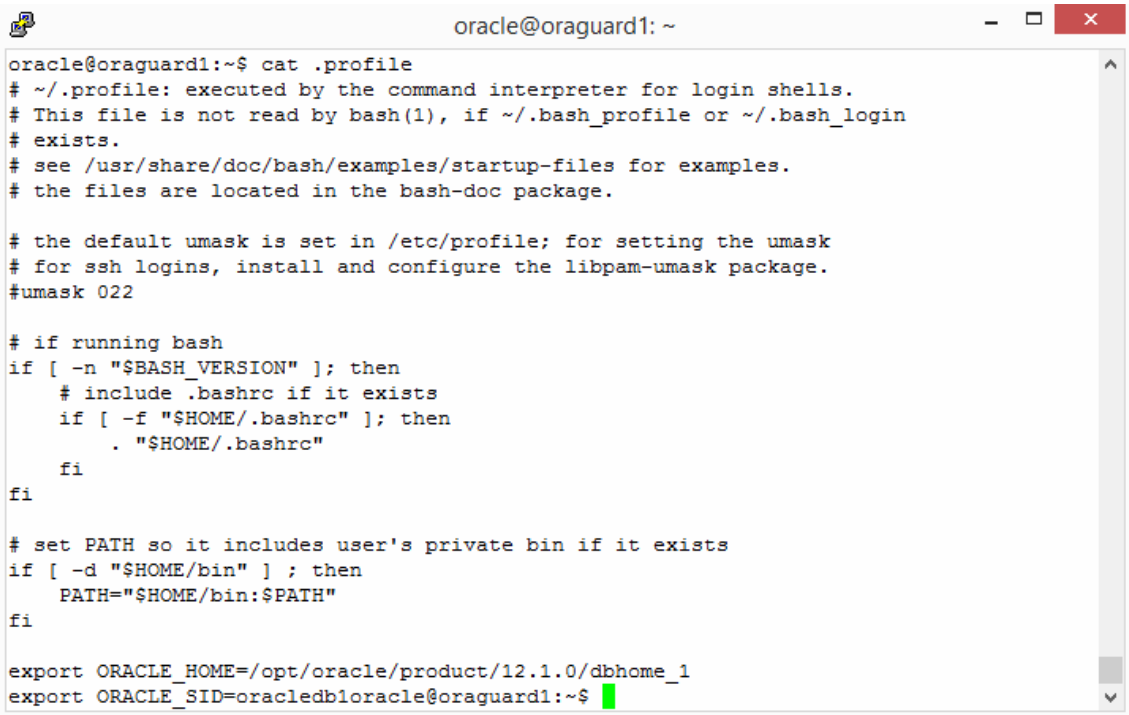
[47] Best Practices for Synchronous Redo Transport. *Oracle* [online]. Redwood Shores: oracle, 2015 [cit. 2018-05-13]. Dostupné z: http://www.oracle.com/technetwork/database/availability/sync-2437177.pdf

[48] Cross-Platform Migration using Heterogeneous Data Guard. *Blogs.oracle* [online]. Toronto: Oracle, 2011 [cit. 2018-05-14]. Dostupné z: https://blogs.oracle.com/upgrade/cross-platform-migration-using-heterogeneous-data-guard

Přílohy

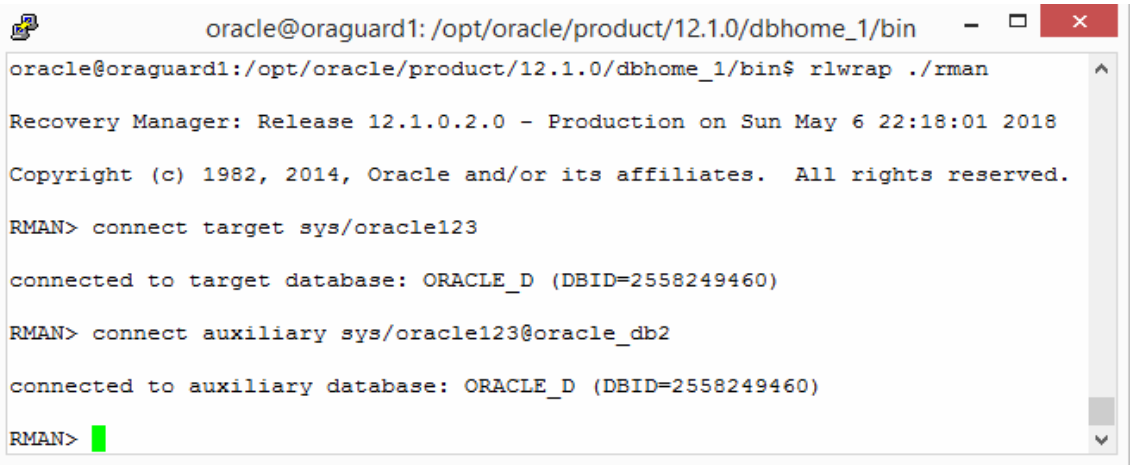
Příloha A - Příprava před instalací

Soubor s parametry uživatele `oracle`. Poslední dva řádky uchovávají parametry `ORACLE_HOME` a `SID`.



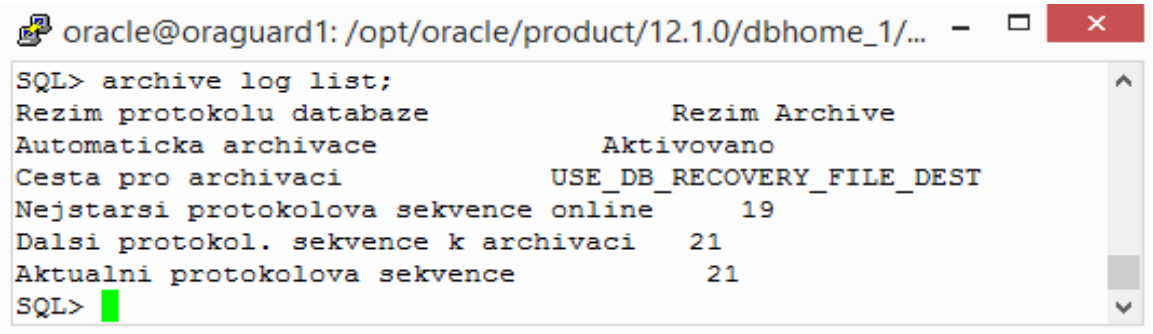
Obrázek 19: .profile - zdroj: vlastní zpracování

Screenshot zaznamenávající připojení k databázím pomocí Recovery Managera.



Obrázek 20: RMAN - zdroj: vlastní zpracování

Výpis ukazující aktivní reğim archivace redo logŢ.



Obrázek 21: reğim archivace - zdroj: vlastní zpracování

Soubor LISTENER.ORA s nastavenými parametry pro ODG na poļítaļi oraguard1 (primární databáze).

```
oracle@oraguard1:/opt/oracle/product/12.1.0/dbhome_1/networ
k/admin$ cat listener.ora
# listener.ora Network Configuration File:
/opt/oracle/product/12.1.0/dbhome_1/network/admin/listener.
ora
# Generated by Oracle configuration tools.

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = oracle_db1)
      (ORACLE_HOME = /opt/oracle/product/12.1.0/dbhome_1)
      (SID_NAME = oracledb1)
    )
   (SID_DESC =
     (GLOBAL_DBNAME = oracle_db1_DGMGRL)
     (ORACLE_HOME = /opt/oracle/product/12.1.0/dbhome_1)
     (SID_NAME = oracledb1)
    )
  )

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST =
oraguard1.zcu.cz)(PORT = 1521))
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
    )
  )

ADR_BASE_LISTENER = /opt/oracle
```

Obrázek 22: LISTENER.ORA na primární databázi - zdroj: vlastní zpracování

Soubor LISTENER.ORA s nastavenými parametry pro ODG na poļítaļi oraguard2 (standby databáze).

```
oracle@oraguard2:/opt/oracle/product/12.1.0/dbhome_1/networ
k/admin$ cat listener.ora

# listener.ora Network Configuration File:
/opt/oracle/product/12.1.0/dbhome_1/network/admin/listener.
ora
# Generated by Oracle configuration tools.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST =
oraguard2.zcu.cz)(PORT = 1521))
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = oracle_db2)
      (ORACLE_HOME = /opt/oracle/product/12.1.0/dbhome_1)
      (SID_NAME = oracledb2)
    )
    (SID_DESC =
      (GLOBAL_DBNAME = oracle_db2_DGMGRL)
      (ORACLE_HOME = /opt/oracle/product/12.1.0/dbhome_1)
      (SID_NAME = oracledb2)
    )
  )
```

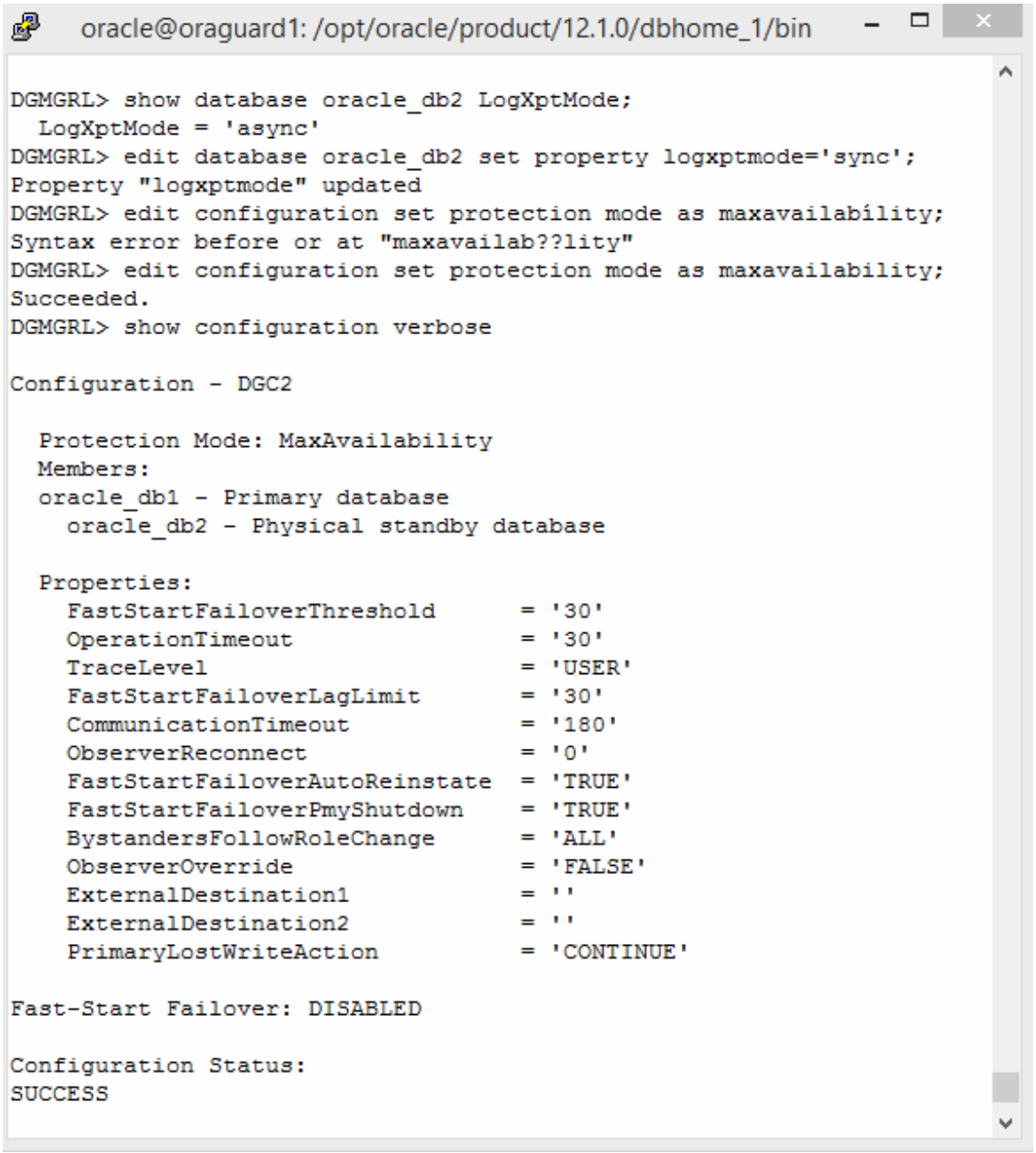
Obrázek 23: LISTENER.ORA na standby databázi - zdroj: vlastní zpracování

Soubor TNSNAMES.ORA s konfigurací pro ODG na poļítaļích oraguard1 (primární databáze) a oraguard2 (standby databáze)

```
oracle@oraguard1:/opt/oracle/product/12.1.0/dbhome_1/networ
k/admin$ cat tnsnames.ora

# tnsnames.ora Network Configuration File:
/opt/oracle/product/12.1.0/dbhome_1/network/admin/tnsnames.
ora
# Generated by Oracle configuration tools.

ORACLE_DB1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST =
oraguard1.zcu.cz)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = oracle_db1)
    )
  )

ORACLE_DB2 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST =
oraguard2.zcu.cz)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = oracle_db2)
    )
  )
```

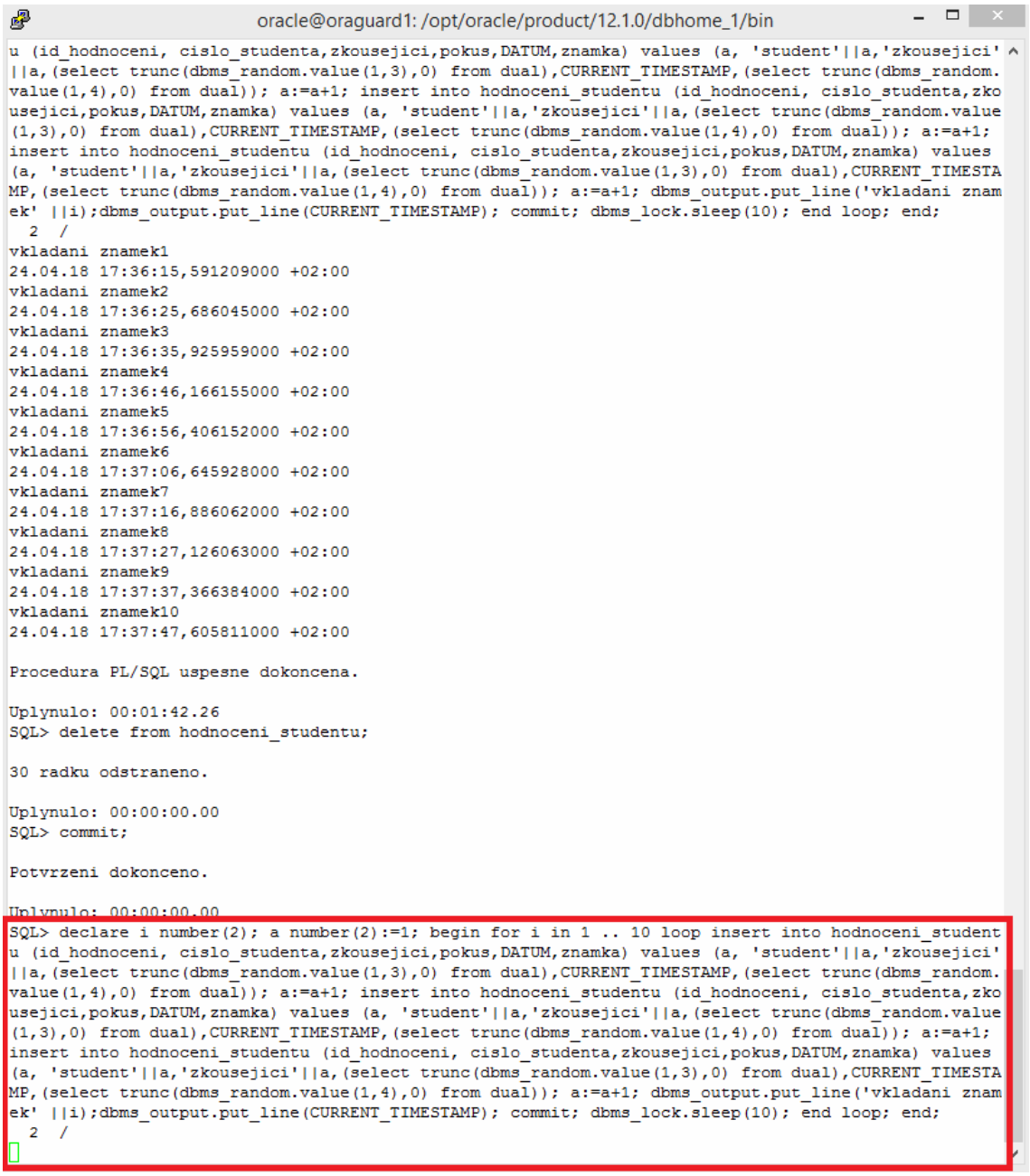
Obrázek 24: TNSNAMES.ORA - zdroj: vlastní zpracování

Výstup DGMGRL zobrazují ODG konfiguraci.

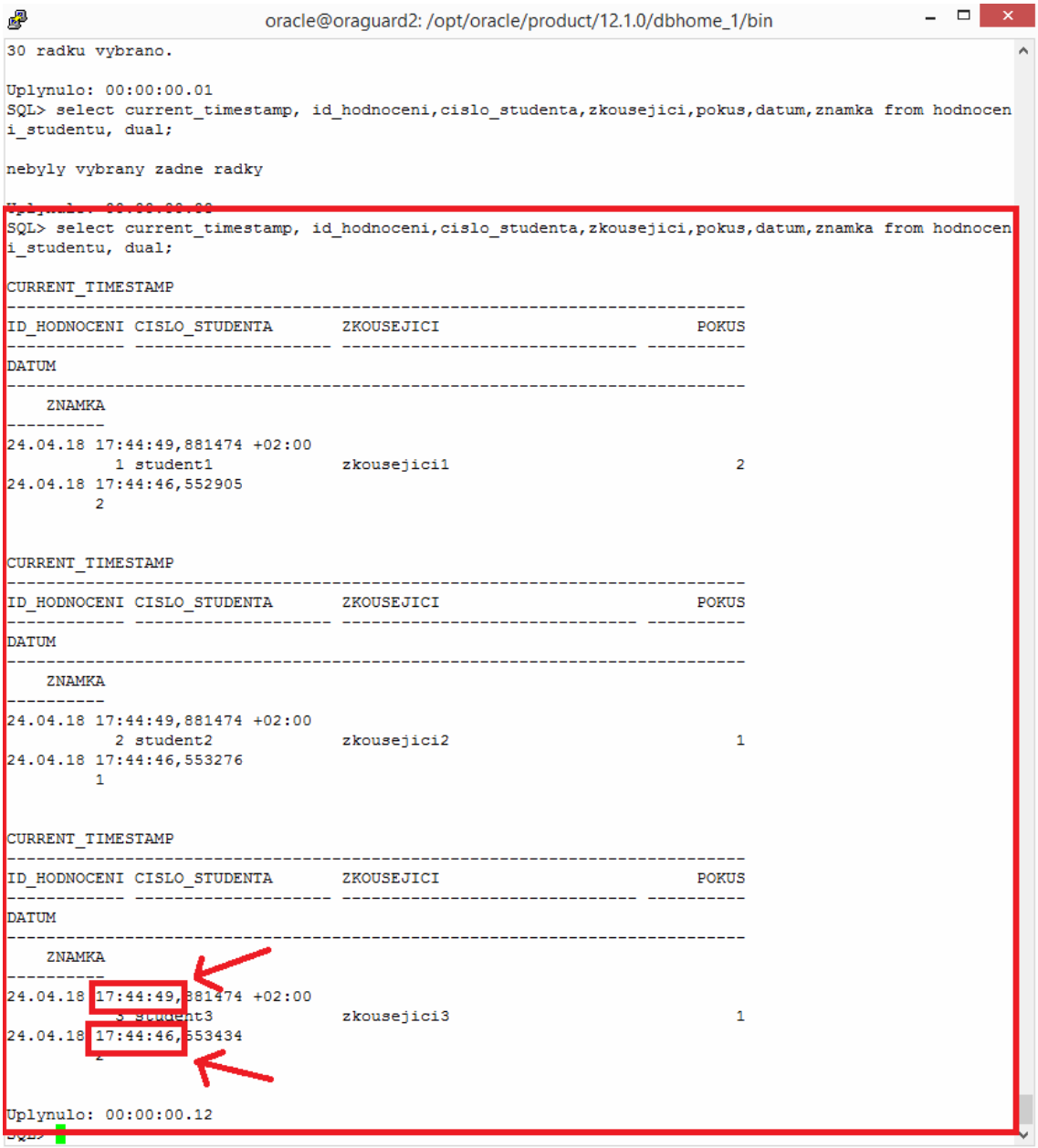


Obrázek 25: ODG konfigurace - zdroj: vlastní zpracování

PŠíloha B: Test standby databáze

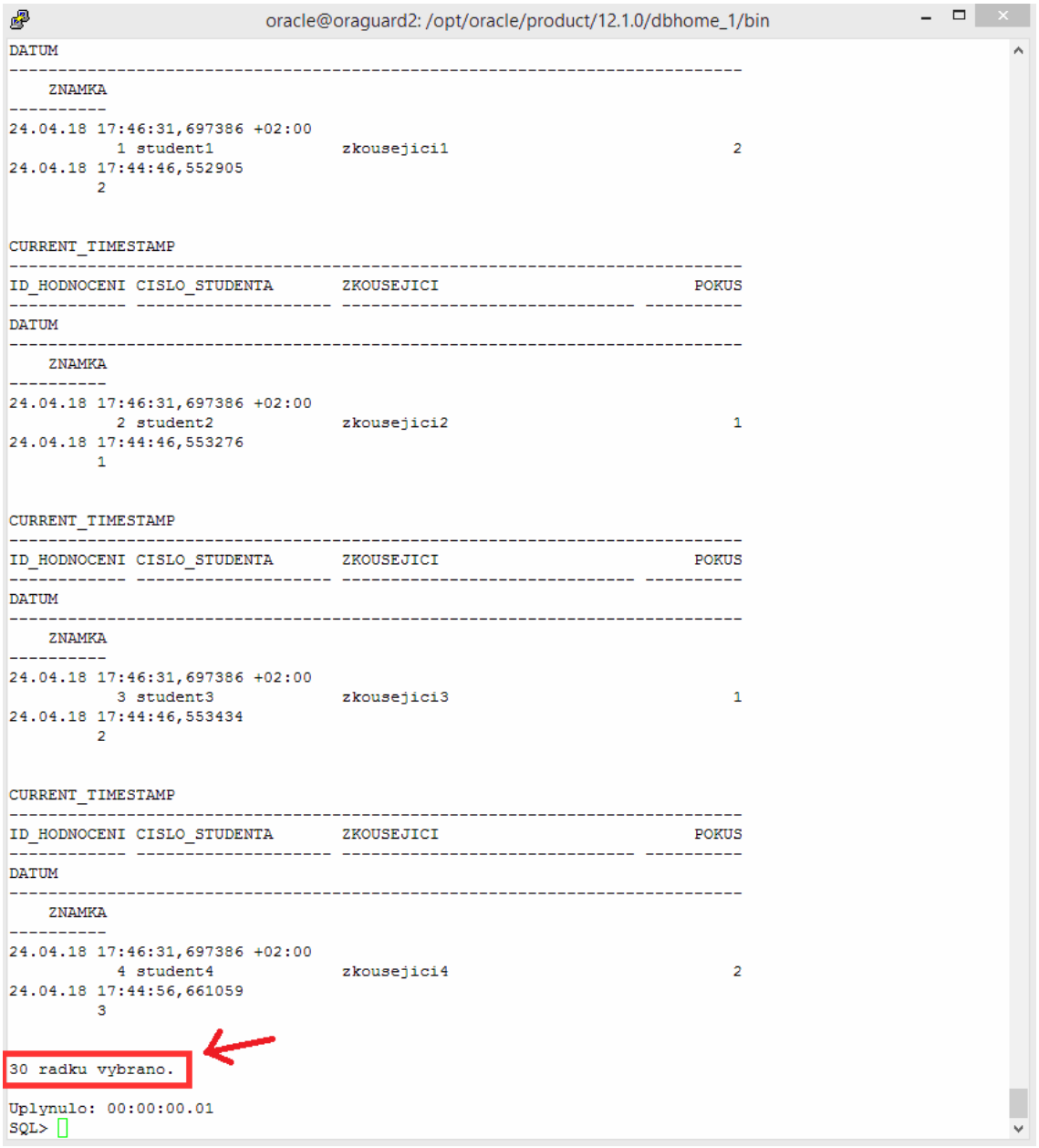


Obrázek 26: test standby databáze - 1. krok - zdroj: vlastní zpracování



Obrázek 27: test standby databáze - 2. krok - zdroj: vlastní zpracování

Obrázek 28: test standby databáze - 3. krok - zdroj: vlastní zpracování

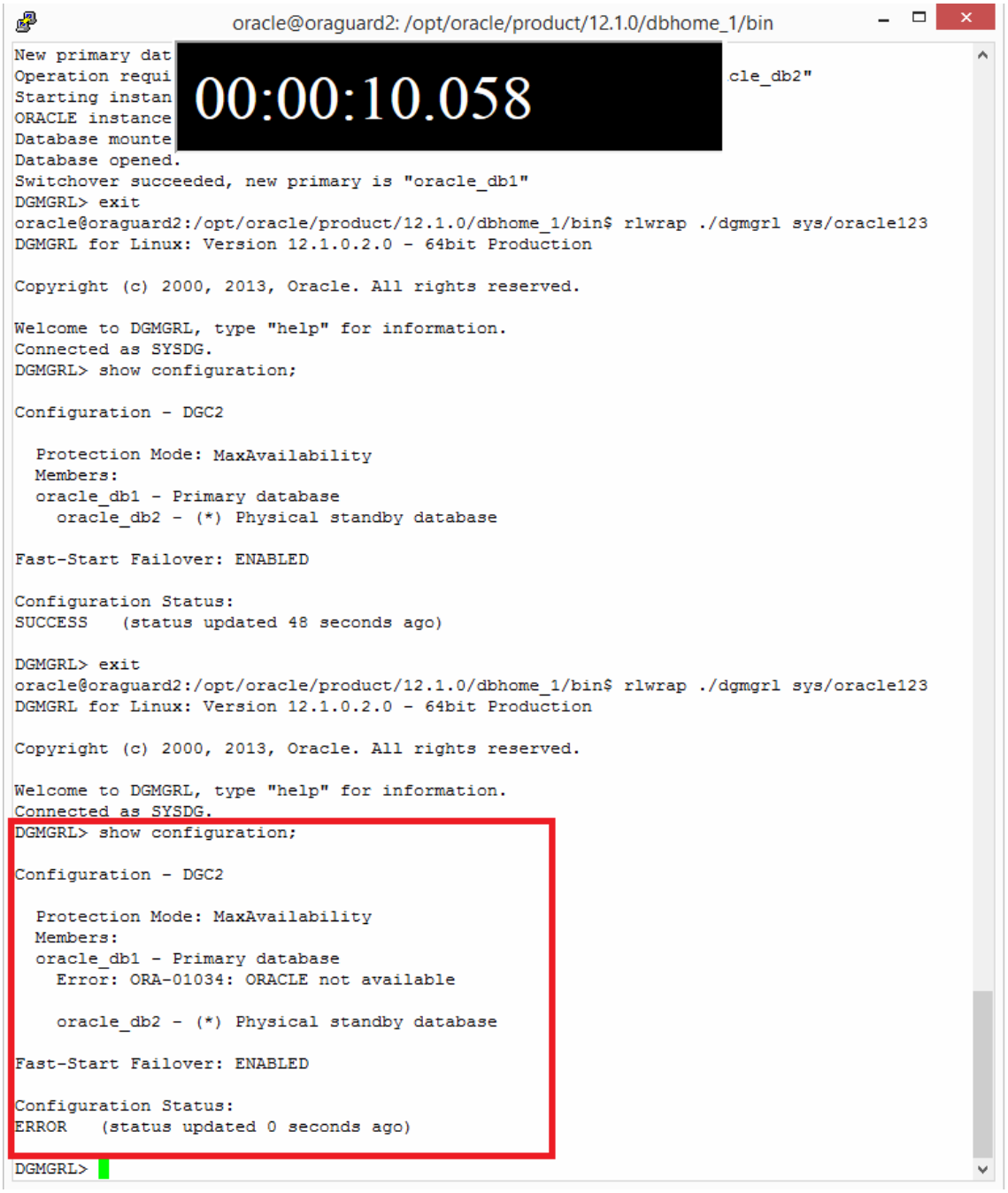


Obrázek 29: test standby databáze - 4. krok - zdroj: vlastní zpracování

PŚíloha C: failover, switchover

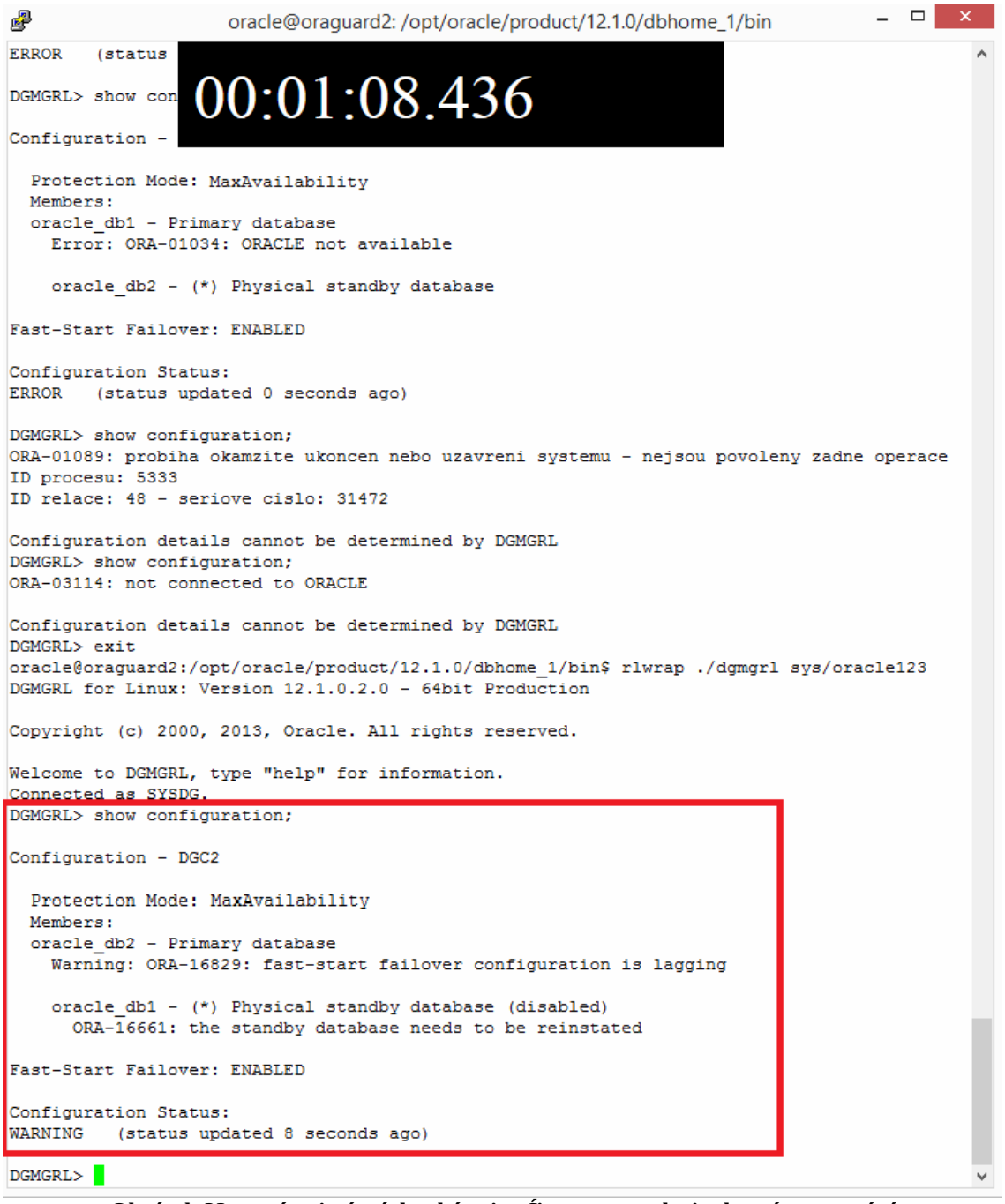
```
# tnsnames.ora Network Configuration File:
/opt/oracle/product/12.1.0/dbhome_1/network/admin/tnsnames.
ora
# Generated by Oracle configuration tools.

ORACLE_DB2 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST =
oraguard2.zcu.cz)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = oracle_db2)
      (INSTANCE_NAME = oracledb2)
    )
  )

ORACLE_DB1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST =
oraguard1.zcu.cz)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = oracle_db1)
      (INSTANCE_NAME = oracledb1)
    )
  )


ORACLE_SERVICE =
  (DESCRIPTION =
    (FAILOVER=ON)
      (ADDRESS = (PROTOCOL = TCP)(HOST =
oraguard1.zcu.cz)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST =
oraguard2.zcu.cz)(PORT = 1521))

    (CONNECT_DATA =
      (SERVICE_NAME = oracle_SERVICE)
    )
  )
```

Obrázek 30: TNSNAMES.ORA pro klient failover - zdroj: vlastní zpracování

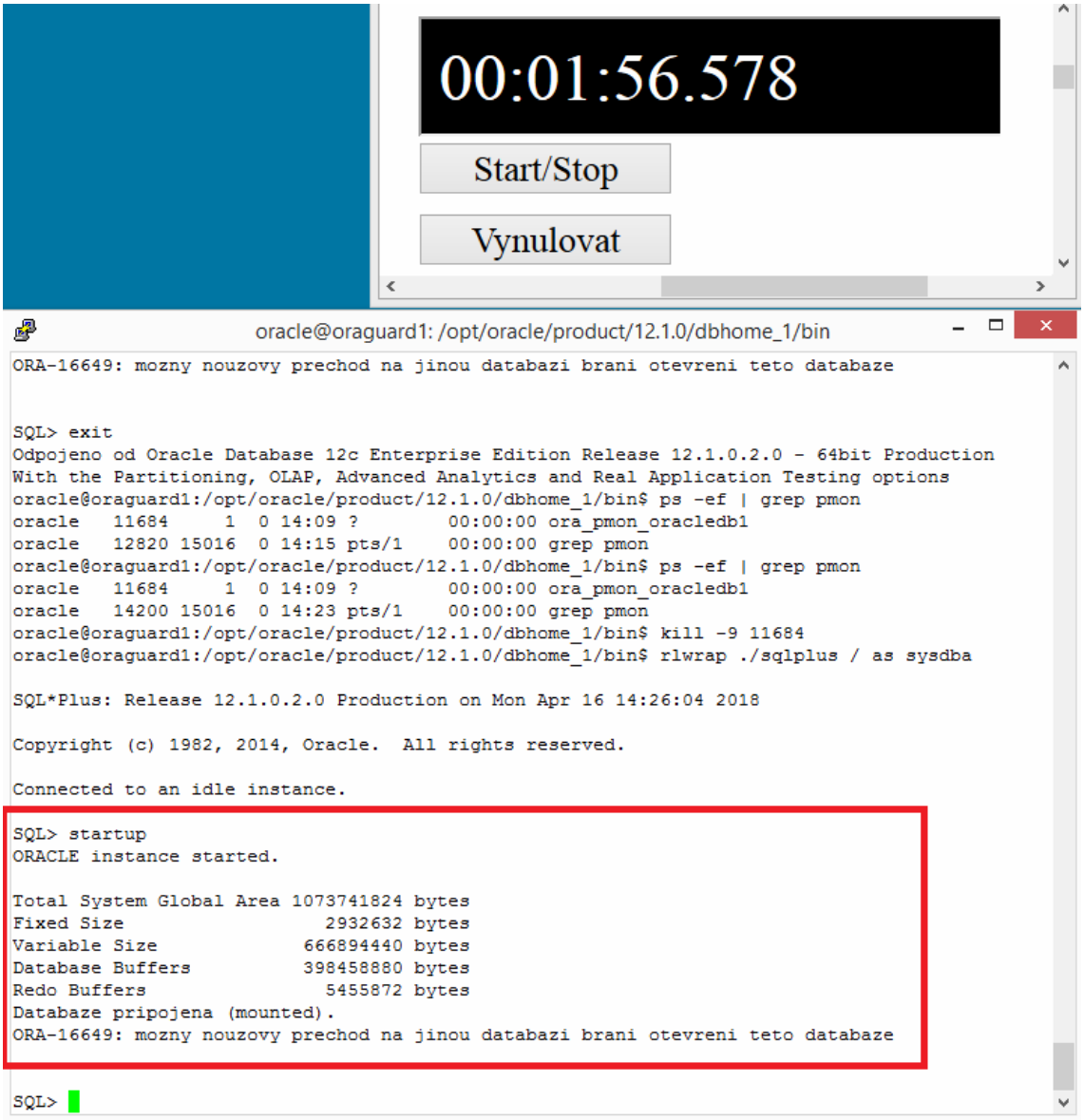


Obrázek 31: výpadek databáze - zdroj: vlastní zpracování

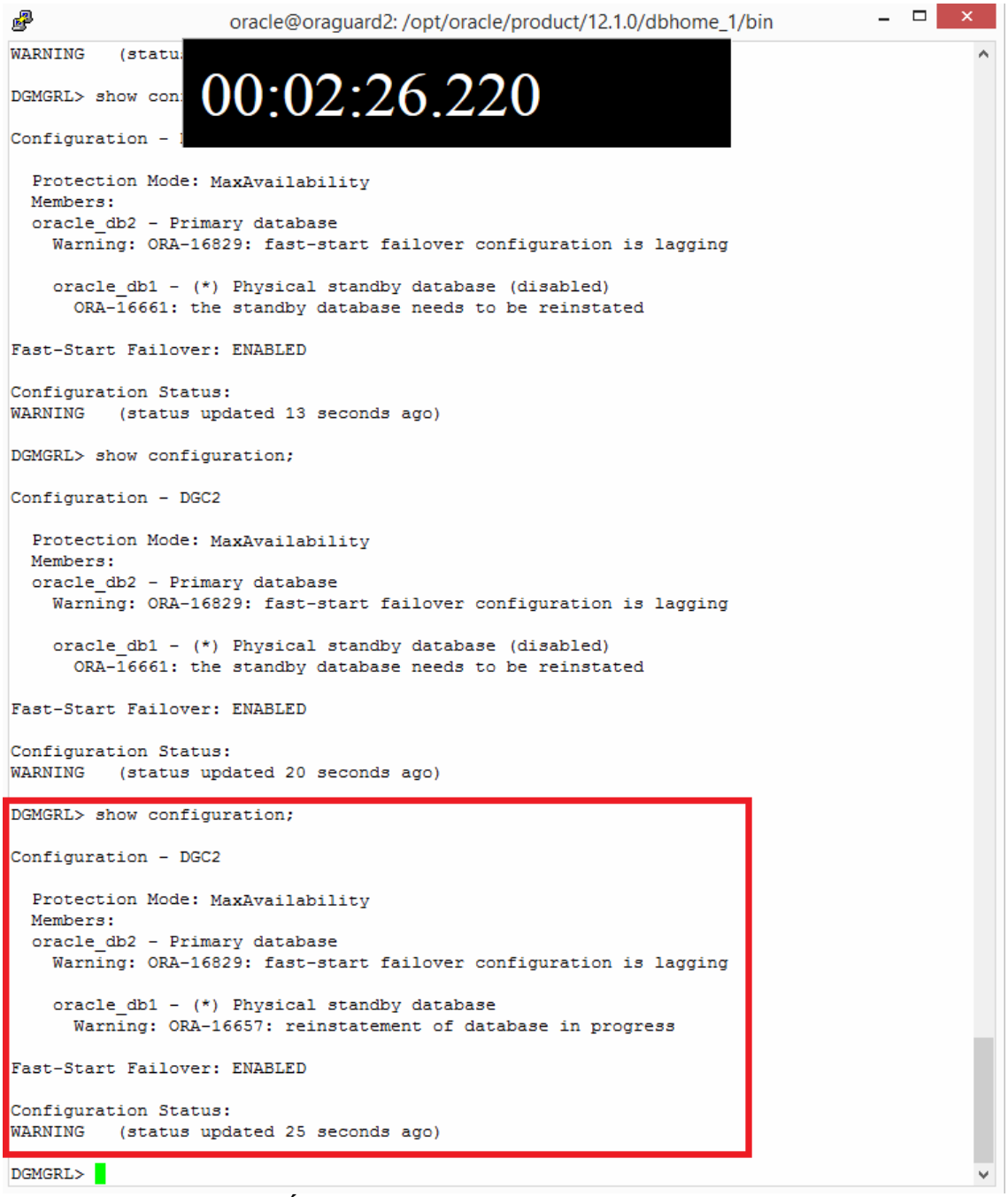
Obrázek 32: zahájení operace failover - zdroj: vlastní zpracování



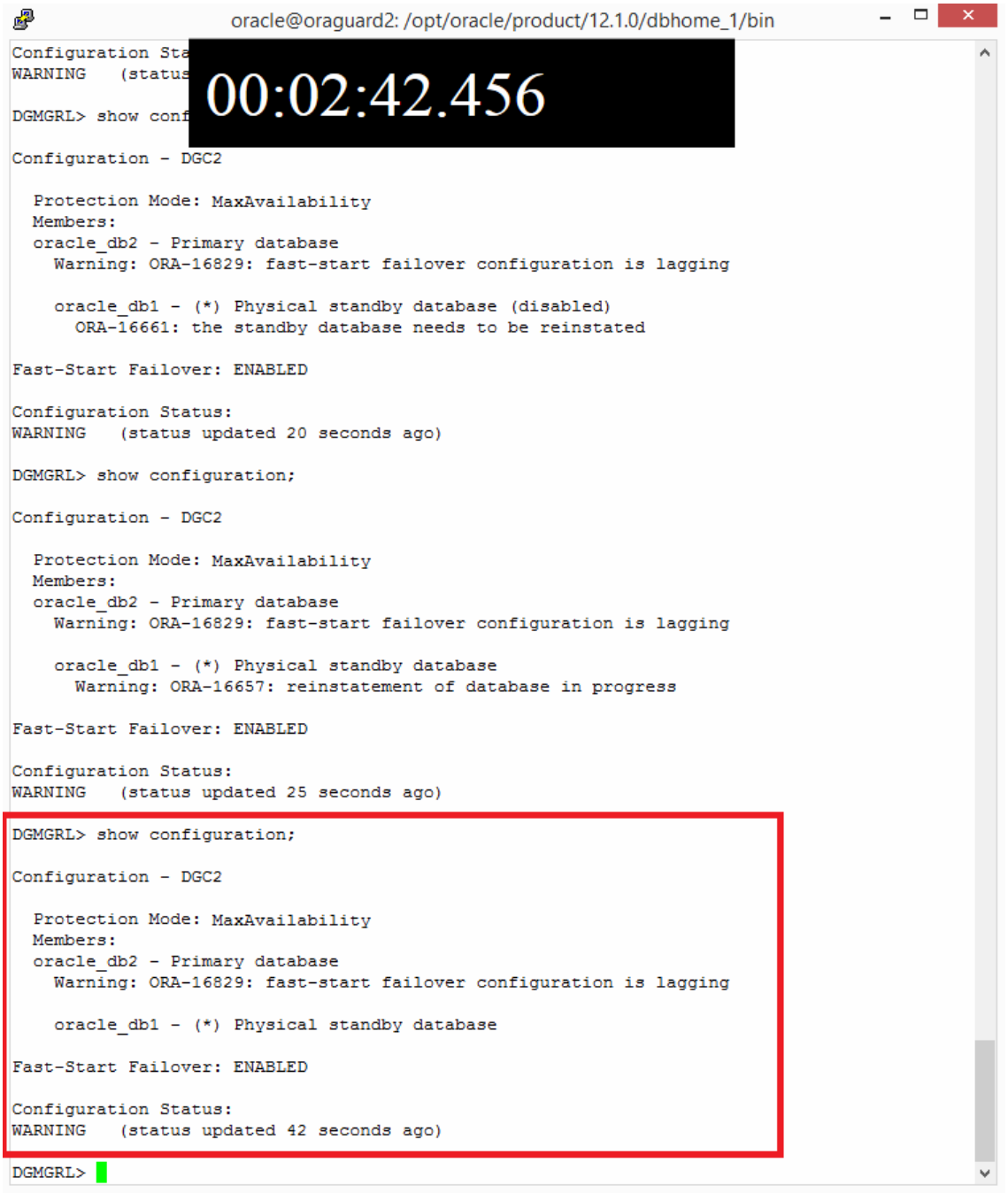
Obrázek 33: nová primární databáze je pŠipravena - zdroj: vlastní zpracování



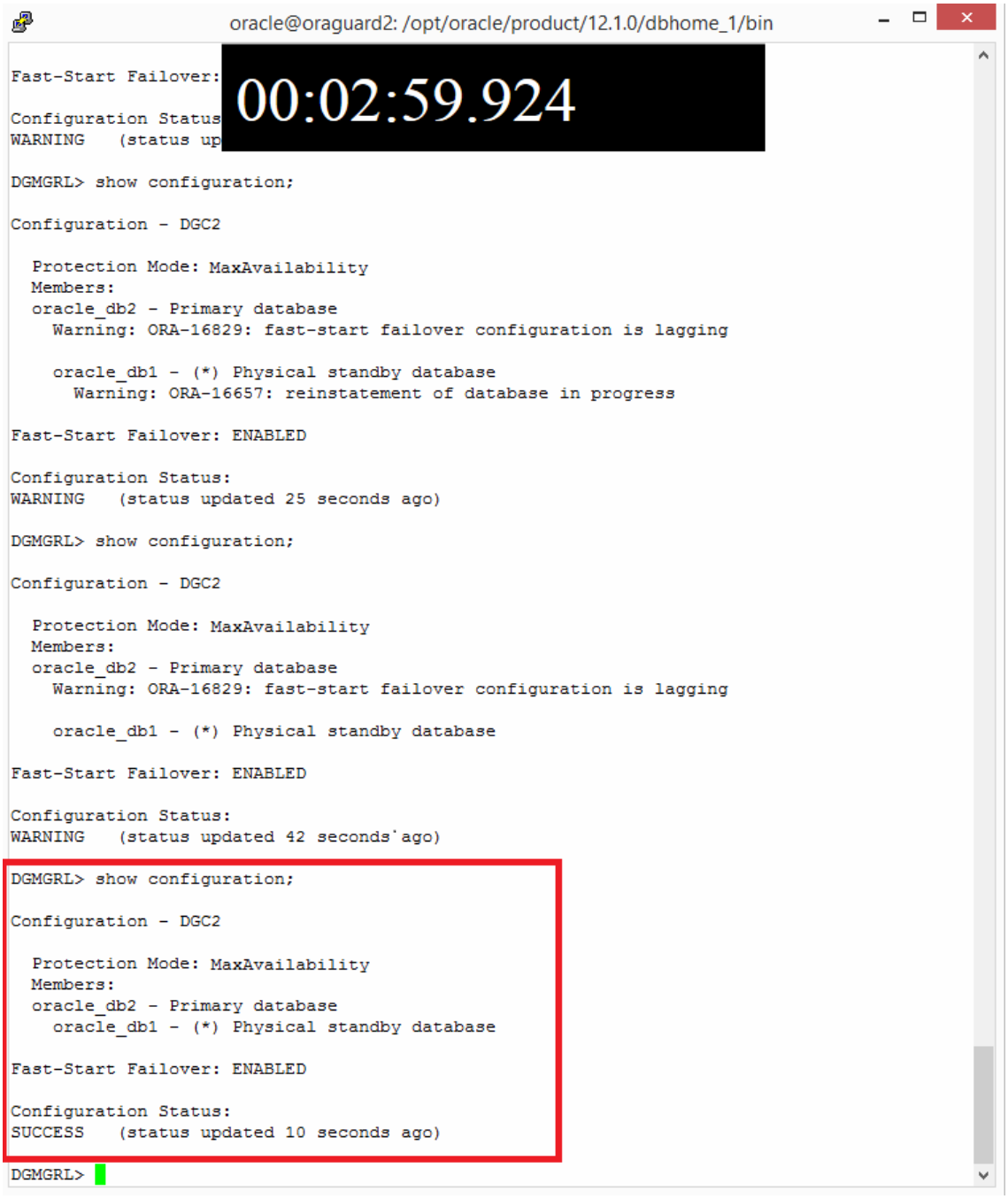
Obrázek 34: obnova databáze po výpadku - zdroj: vlastní zpracování



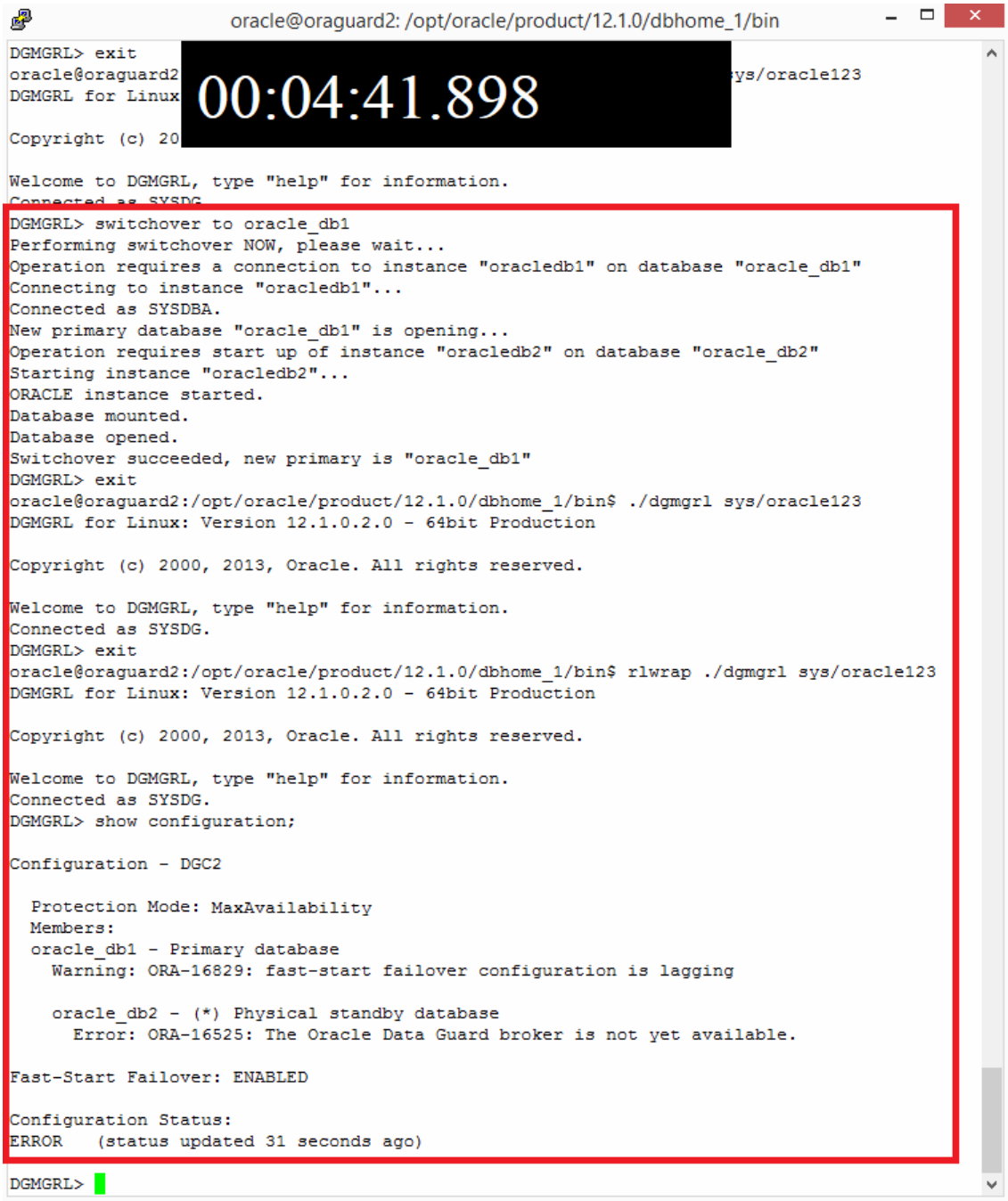
Obrázek 35: pŠíprava nové standby databáze - zdroj: vlastní zpracování



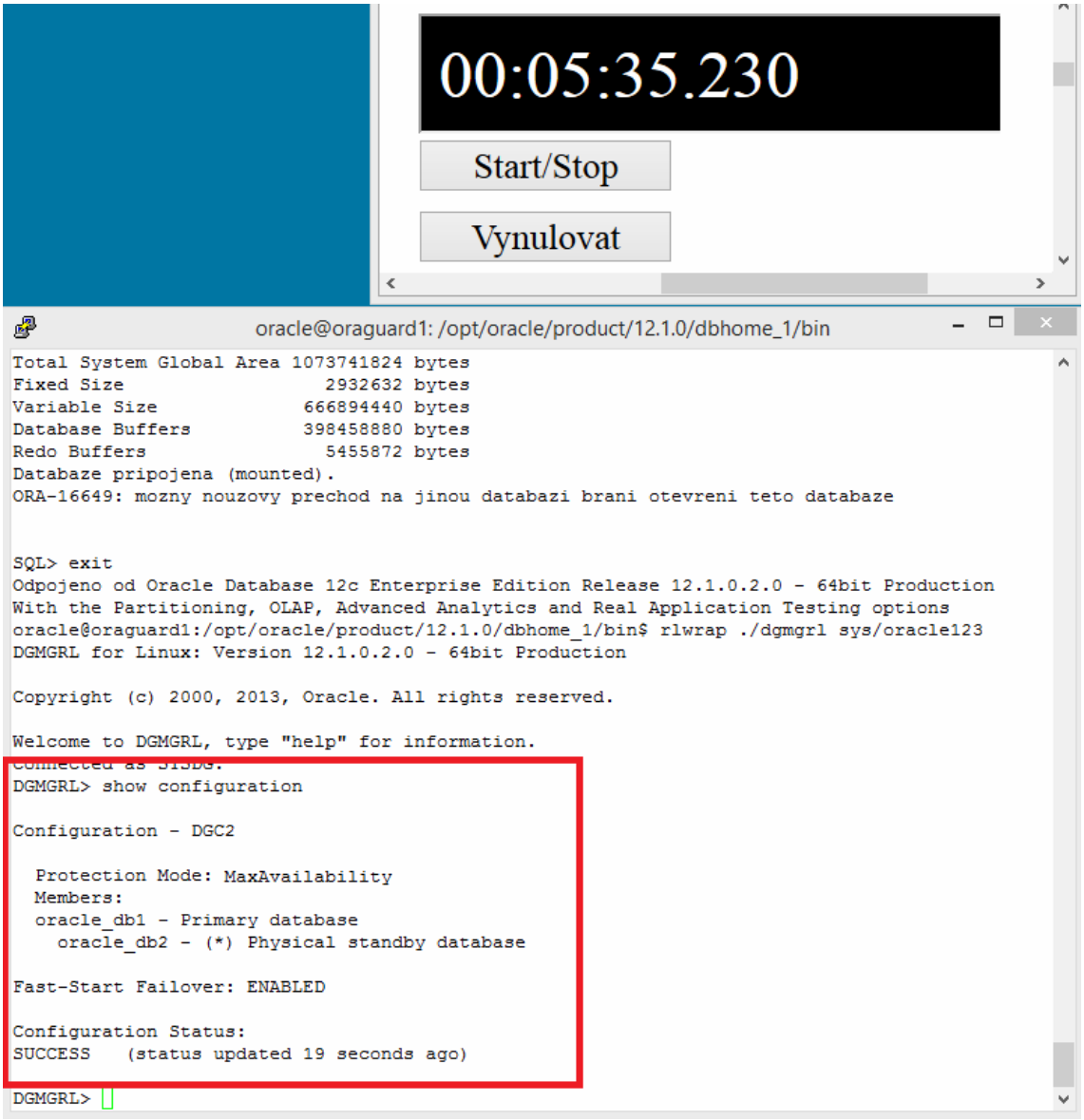
Obrázek 36: nedokonļená synchronizace - zdroj: vlastní zpracování



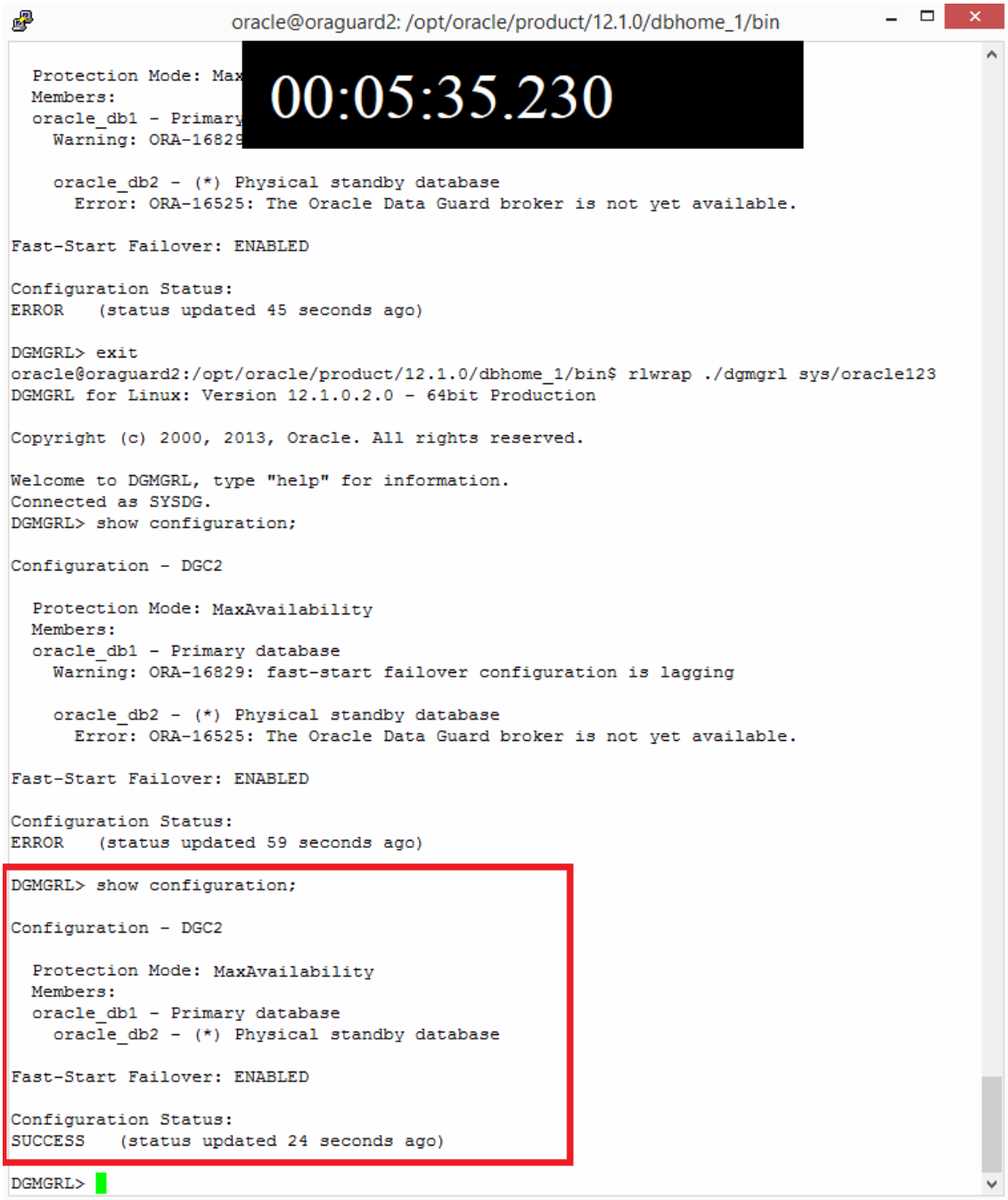
Obrázek 37: nová plnĎfunkļní ODG konfigurace - zdroj: vlastní zpracování



Obrázek 38: switchover - zdroj: vlastní zpracování



Obrázek 39: oraguard1- funkční ODG konfigurace - zdroj: vlastní zpracování



Obrázek 40: oraguard2 - funkční ODG konfigurace - zdroj: vlastní zpracování

Mě̌ení	Odchylky	Ų tverce odchylek		
342	55,65	3096,92		
252	34,35	1179,92		
272	14,35	205,92		
272	14,35	205,92		
285	1,35	1,82		
340	53,65	2878,32		
261	25,35	642,62		
307	20,65	426,42		
266	20,35	414,12		
239	47,35	2242,02		
336	49,65	2465,12		
277	9,35	87,42		
290	3,65	13,32		
256	30,35	921,12		
286	0,35	0,12		
276	10,35	107,12		
286	0,35	0,12		
344	57,65	3323,52		
270	16,35	267,32		
270	16,35	267,32		
PrŢmČr	PrŢmČr	Rozptyl	SmČrodatná odchylka	PŘípustná chyba
286,35	24,09	986,66	31,41	13,77

Tabulka 6: výpoļty k testŢm - zdroj vlastní zpracování