Budapesti Corvinus Egyetem

Gazdálkodástudományi Kar

Információrendszerek Tanszék

Üzletfolytonosság-menedzsment Oracle alapokon

Magas rendelkezésre állású adatbázisok megvalósítása

Készítette: Kóródi Ferenc

Gazdaságinformatikus (MSc) szak

2011

Szakszeminárium-vezető: Mohácsi László

Tartalomjegyzék

1. Előszó

A szakdolgozat témaválasztáshoz az elmúlt hónapokban tanultak és tapasztaltak nyújtottak elsődleges támpontot. Ez idő alatt – gyakornokként – sikerült részletesebben megismerkednem hazánk egyik legnagyobb bankjának adatbázis-rendszereivel és az általános adatbázis-adminisztrátor feladatköreivel. Betekintést nyerhettem az Oracle adatbázis-kezelő rendszer nyújtotta szolgáltatásokba és lehetőségekbe, amelyek egy nagyvállalat – esetemben egy bank – számára kiemelt fontossággal bírnak. Ezeket a tapasztalatokat szeretném felhasználni szakdolgozatom elkészítéséhez.

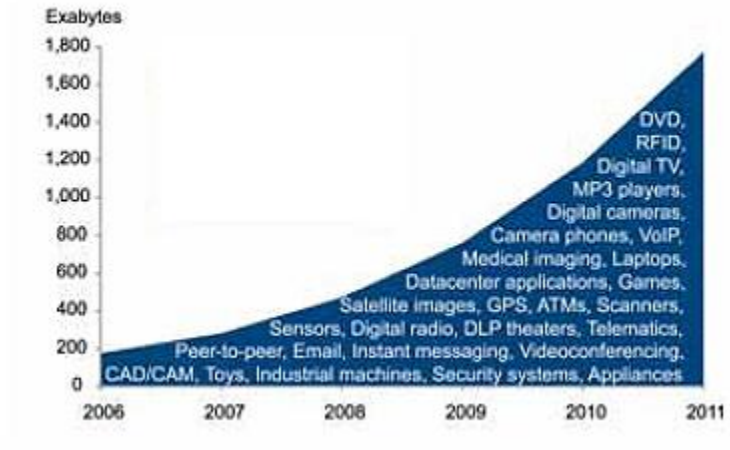
Minden vállalatnak szüksége van olyan megoldásokra, amelyek segítségével képes az esetleges adatvesztésből, az adatközpont üzemzavarából vagy természeti katasztrófákból adódó problémák elhárítására. Ezek orvoslására szolgál átfogó néven az üzletfolytonosság-, és rendelkezésre állás menedzsment. Meg kell említenem a felhasználók által jelentett kockázatokat is, amelyeket szintén a megfelelő intézkedésekkel kell kezelni, valamint a kontrollálhatóság érdekében nyomon követni.

Szakdolgozatomat 3 elkülöníthető, de egymásra épülő részből áll össze. Az első rész az elméleti bevezetést tartalmazza, amely az üzletfolytonosság menedzsmentet, a különböző veszélyhelyzeteket megelőző, vagy kezelő terveket, mint például a DRP vagy a BCP jelenti. Itt esik szó többek között a stratégiai tervezésről, a kockázat menedzsmentről is. Az ezt követő rész azokról az Oracle megoldásokról szól, amelyek biztosítják az első pontban leírtak gyakorlati megvalósítását. Röviden összefoglalva a standby adatbázisokról, a mirror rendszerekről, az ezek megvalósítását biztosító DataGuard szolgáltatásról lesz szó, valamint az Oracle RMAN-ról, amely a recovery management-et teszi lehetővé, és ennek funkcióiról. Röviden bemutatom majd az auditálási lehetőségeket is, amelyek biztosítják a felhasználók nyomon követhetőségét és kontrollálását. A harmadik részben a gyakorlati bemutatásra kerül sor, ahol ismertetem, hogy a gyakorlatban hogyan is történik egy olyan adatbázis-kezelő rendszer megvalósítása, amely kellőképpen megfelel az üzleti elvárásoknak, és az üzletmenet-folytonossági igényeknek.

2. Bevezetés

2.1. Az adatok értéke

Információs társadalom és tudásgazdaság. Manapság egyre gyakrabban használjuk ezeket a kifejezéseket. Az átalakuló gazdasági és társadalmi folyamatok körében az információnak és az abból származó tudásnak egyre nagyobb szerepe van. Ez a gazdasági élet minden területére igaz. Aki a stratégiailag fontos információt birtokolja, az versenyelőnyhöz jut a vetélytársakkal szemben. Az információk megszerzésének és feldolgozásának módja tehát fontos szereppel bír. A vállalatok szempontjából a tudás, mint vállalati érték kritikus fontosságú elemmé lépe elő. Az adatok, mint az információ és a tudás alapja ma már felfoghatatlan mennyiségben keletkeznek és tárolódnak el különböző informatikai rendszerekben. Az alábbi ábra szemlélteti, hogy milyen dinamikusan nő a keletkezett adatok mennyisége.



1. ábra: Digitális információ keletkezése világszerte (*Forrás: IDC, 2008*)

Az üzleti szektort tekintve az adatok értéke még inkább felértékelődik. Az adatok és az abból származó tudás a kiélezett versenyhelyzetben kulcsfontossággal bírnak. Az adatbázis-kezelő rendszerek és az azokhoz kapcsolódó egyéb üzleti informatikai rendszerek, mint például az üzleti intelligencia rendszerek, a tudásmenedzsment rendszerek ma már minden tudásalapú vállalatnál kiemelt szerepet töltenek be. Ebből következik, hogy az adatok védelmét és biztonságát, valamint a kritikus informatikai rendszerek magas rendelkezésre állását garantálnia kell a vállalatoknak. A cél az, hogy elkerüljék a hardver meghibásodásból, az alkalmazáshibákból, katasztrófákból származó veszélyeket. Emellett még a felhasználói eredetű hibákat is érdemes

megemlíteni. Továbbá az SLA-k (Service Level Agreement) – magyarul a szolgáltatási szint szerződés – teljesítése is alapvető elvárás az informatikai rendszerek üzemeltetőitől. Ebben a szolgáltatási szint szerződésben állapodik meg a szolgáltató és az ügyfél a nyújtott szolgáltatás szintjéről. Ez magában foglalja a szolgáltatás leírását, a maximális kiesési időt, a rendszer felelőseit, és a rendelkezésre állás egyéb paramétereit.

Az előbbiekben felsorolt kihívásokra az üzletfolytonosság-menedzsment képes válaszokat nyújtani.

2.2. Üzletfolytonosság-menedzsment

Az üzleti folyamatok nagy része jelentős mértékben függ az alkalmazott technológiáktól és az informatikai rendszerektől. Ezeket a rendszereket különböző fenyegetettségek érhetik. Fenyegetettségnek tekintünk minden olyan eseményt, amely megzavarja, adott esetben pedig akár lehetetlenné is teszi a normális működést. Ennek kockázatait mérlegelni kell, megelőzésére pedig megelőző lépéseket kell tenni. A folytonosság fenntartása pedig nagymértékben függ a vezetők hozzáállásától, az előzetes felmérésektől és megelőző lépésektől. A legfontosabb cél a legjelentősebb fenyegetettség, a katasztrófa elkerülése. Katasztrófa alatt érjük az előre nem tervezett olyan eseményt, vagy hatást, amely megszakítja vagy megakadályozza akár rövidebb időre is a kritikus folyamatok végrehajtását. Az üzleti-folytonosság tervezés több okból kifolyólag is fontos. Akár törvényi kötelezettségnek is eleget tehet, a károk veszteségét minimalizálhatja, költségmegtakarítást eredményezhet, üzleti igényeknek felelhet meg. (*Raffai, 1999*)

Az üzletmenet-folytonosság (Business Continuity Management – BCM) tehát a kockázatmenedzsmenttel van szoros kapcsolatban, tulajdonképpen annak eredményének is tekinthető. Célja, hogy felkészülést biztosítson arra az esetre, ha a kritikus üzleti folyamatok sérülnek, vagy teljes egészében leállnak. Az egészen kicsitől a teljes kiesésig terjedő leállásokat is figyelembe kell venni. Az üzletmenet-folytonossági koncepciók kialakítása a vezetőség feladata, az ő elkötelezettségük kell a hatékony megvalósításhoz. A BCM magába foglalja a szükséges dokumentumok, tervek elkészítését, a tesztelést, az oktatást, valamint a karbantartást is. A vállalat

biztonságpolitikájának megfelelő katasztófatervet valósíthat meg az üzletmenet-folytonosság tervezés (Business Continuity Planning - BCP) során. A BCP magába foglalja a kockázatelemzést, az üzleti hatáselemzést, a megelőző lépések tervezését, a helyreállítási stratégia kialakítását, a lehetséges károk meghatározását, a megfelelő dokumentációk elkészítését és mindezek ellenőrzését és felülvizsgálatát. Érdemes kiemelni az üzleti hatáselemzést (Business Impact Analysis – BIA) egy kicsit, aminek célja, hogy a vezetőség számára összefoglalja a nem kívánt események hatásait, és lehetséges következményeit. Tartalmazza a folyamatok prioritását, és a leállási idők elemzését, valamint becslést nyújt a kiesés következtében keletkező károkra. Lépései a következők: információgyűjtés az üzleti területek vezetőit bevonva interjúk keretében, az információk elemzése, a különböző mutatószámok kialakítása és megállapítása. A hatáselemzés legfontosabb feladata, hogy az egyes folyamatokat prioritási csoportokba sorolja, annak érdekében, hogy megállapítható legyen, mely folyamatokkal kell kiemelten foglalkozni és részletes tervezést biztosítani. Fontos a vállalati folyamatok feltárása és elemzése, valamint ezáltal a kritikus folyamatok azonosítása. Csak akkor lehetséges a megfelelő tervek kialakítása, ha a vállalat tisztában van saját folyamataival és azok összefüggéseivel. Az üzleti folyamatok feltárása (Process Mapping) során azonosított kritikus folyamatok kezelése nélkülözhetetlen a zavartalan működés érdekében. (*Raffai, 1999*)

Az üzletfolytonosság-menedzsment feladatai közé tartozik a különböző tervek kialakítása, tesztelése, oktatása és felülvizsgálata. Ide sorolható a BCP cselekvési tervek, a katasztrófa helyreállítási-, (Disaster Recovery Plan – DRP), a tesztelési-, és az oktatási tervek. A BCP cselekvési tervek kimondottan az üzleti hatáselemzésre épülnek. Azt határozzák meg, hogyan lehet a katasztrófa idején az adott folyamatot fenntartani. A DRP tervek a legrosszabb esemény bekövetkezése utáni helyreállítási feladatokról gondoskodnak. Meghatározzák a helyreállítás módját, ütemezését, valamint becslést adnak a lehetséges károkra is. Az egész tervezés mit sem ér, ha azt a felhasználók nem ismerik, nem tudják mi a teendő akkor, ha bekövetkezik a katasztrófa. Ezért szükséges az oktatás és a tesztelés megvalósítása is (*Raffai, 1999*).

2.3. Az Oracle jelenlegi piaci helyzete

Az Oracle az egyik meghatározó IT céggé nőtte ki magát az évek folyamán köszönhetően sikeres vállalati stratégiájának. A Fortune magazin 2010-es felmérése alapján az eladásokat tekintve a világ 366. legnagyobb bevételét produkáló vállalata lett. Ma már piacvezető cégként lehet rá tekinteni az adatbázis-kezelő szoftverek piacán. Az alábbi táblázat is ezt támasztja alá. Látható, hogy a kétezres évek közepén az adattárházak piacának majdnem harmadát birtokolta az Oracle egymaga.

Vállalat	2005 (Millió $)	2006 (Millió $)	2007 (Millió $)	2005 Részesedés (%)	2006 Részesedés (%)	2007 Részesedés (%)
Oracle	1653,2	1885,4	2136,7	31,6	32	31,7
IBM	1120,7	1254,4	1423,7	21,6	21,4	21,2
Microsoft	630,2	775,7	904,7	12,1	13,3	13,5
Teradata	523,8	538,3	609,7	10,1	9,2	9,1
SAS	457,4	498,7	548,9	8,5	8,5	8,2

1. táblázat: Az adattárházak piacának árbevételi adatai (*Forrás: IDC, 2008*)

Az Oracle Database szoftver ma is a legmeghatározóbb terméke a vállalatnak, azonban az Oracle ma már több mint szoftverfejlesztő vállalat. A termékek és szolgáltatások széles palettáját képesek nyújtani a világ bármely pontján, a nap bármely órájában. Ennek oka a kettős növekedési stratégia: a saját fejlesztések megmaradtak és fontos szerepet töltenek be, mint például a Grid Computing, az üzleti intelligencia vagy pedig a Fusion Middleware megoldások. Azonban egyre nagyobb hangsúlyt fektetett a vállalat a felvásárlásokon keresztül történő növekedés. Ma már hetente vásárol fel kisebb vállalatokat és integrálja azok tevékenységét az összvállalat tevékenységébe. Néhány példa a felvásárlásokra:

- 2004: PeopleSoft
- 2005: TimesTen
- 2006: Siebel, Sigma Dynamics
- 2009: Sun Microsystem

A Sun felvásárlása igen meghatározó volt az Oracle történetében. Ekkor kezdett meg a hardver megoldások felé igazán nyitni a vállalat. Nem is kellett sokáig várni az első Oracle szoftverekre optimalizált hardverek – elsősorban szerverek – megjelenésére. Az Oracle adatbázis-kezelő rendszerhez kifejlesztett Exadata szervereket a ma elérhető leggyorsabb adatbázis-szervereknek hirdeti az Oracle, nem is alaptalanul. A Sun

tudásának és az Oracle tudásának kombinálása lehetőséget ad az Exadatához hasonló technológiai megoldások kifejlesztéséhez, ezáltal pedig a piaci pozíció javítását is biztosítja. A vállalatfelvásárlások tehát komoly célokkal szolgálnak. Nem csupán a vállalati érték növelése a cél, hanem a stratégiailag fontos tudás elemek megszerzése és integrálása a vállalatba. Az termékportfólión is látható mindez.

2. ábra: Oracle termékportfólió (*Forrás: www.oracle.com/hu/index.html*)

3. Az Oracle által nyújtott megoldások

Ebben a fejezetben bemutatásra kerülnek azok a legfontosabb Oracle megoldások, amelyek az adatbázis-kezelő rendszer (Oracle Database 11g) szempontjából fontosak, ha üzletfolytonosság-menedzsmentről és magas rendelkezésre állásról van szó. Ezek a megoldások biztosítják, hogy az üzletmenet zavartalanul folytatódhasson, ha tervezett vagy nem tervezett leállásra kerül sor.

3.1. Áttekintés

Az Oracle adatbázis-kezelő rendszer kereskedelmi verzióban 1979-től elérhető a piacon. Ez az első relációs adatbázis, amely az SQL nyelvet használta. A rendszer az elmúlt több mint 40 év alatt folyamatosan változott, igazodva a piaci igényekhez, felhasználói elvárásokhoz. Ma már piacvezető adatbázis-kezelő megoldásként tekinthetünk rá, amely funkcionalitásában sokkal többet kínál, mint egy hagyományos relációs adatbázis-kezelő rendszer (Relational Database Management System - RDBMS). Olyan funkciókat integráltak az évek folyamán a szoftverbe, amelyek ma már alapfunkciókká váltak, és el sem tudnánk képzelni a működési folyamatainkat nélkülük. Példaként ma már a 11gR2-es verzió esetén beszélhetünk Data Mining funkcióról, ami adatbányászati megoldásokat kínál, Real Application Testing-ről, Automatic Storage Management-ről, OLAP funkciókról, Oracle Enterprise Management-ről, Data Warehouse megoldásokról, és még hosszasan sorolhatnám. A vállalatok dönthetnek arról, hogy mely funkciók szükségesek számukra, ugyanis az Oracle RDBMS szoftver többféle konfigurációban elérhető (Express Edition, Personal Edition, Standard Edition One, Standard Edition, Enterprise Edition).

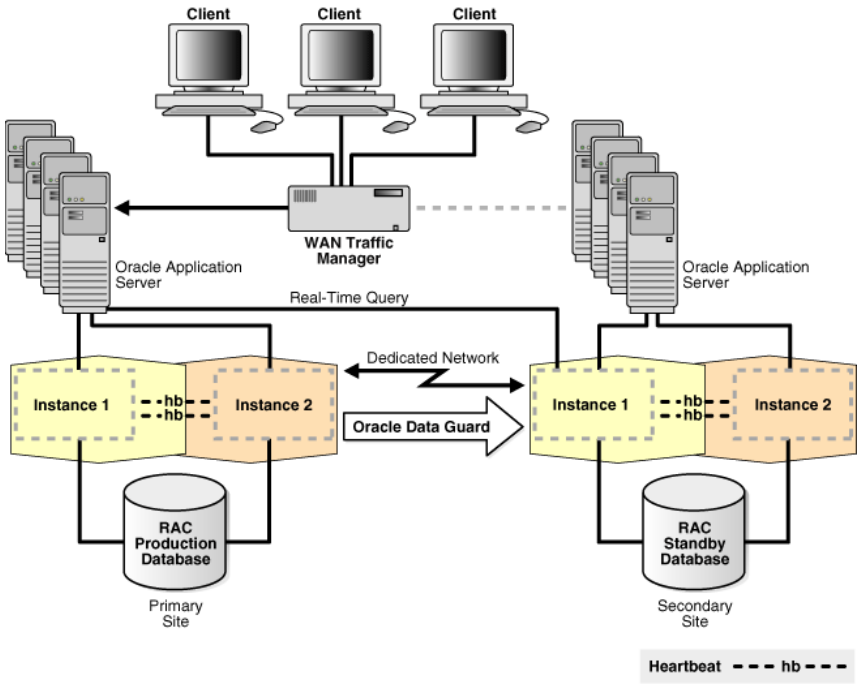
A dolgozatom szempontjából azok a szolgáltatások érdekesek, amelyek a rendelkezésre állást, az üzletmenet-folytonosságot, az adatok védelmét biztosítják. Olyan funkciók ezek, amelyek ma már elengedhetetlenek komoly adatbázisokkal rendelkező vállalatok esetén, ahol az adatbáziskezelő-rendszer leállása beláthatatlan következményekkel járna – gondolok itt pénzben kifejezhető veszteségre, üzleti adatok elvesztésére, vagy akár áttételesen a fogyasztók bizalmának elvesztésére. Gyakorlati példaként felhozható az a kérdés, hogy mi történne akkor, ha egy bank adatbázis-rendszere csupán egy órára elérhetetlenné válna. Az ügyfelek nem tudnának a

pénzükhöz jutni, az alkalmazottak nem végezhetnék napi feladataikat, hiszen minden üzleti folyamat alapvetően az adatbázis rendszerre épül alapvetően. Tehát súlyos presztízs-, és pénzveszteséggel járna mindez a bank számára, amely számára egyszerűen nem megengedhető.

A következő fejezetekben az üzletfolytonossághoz, magas rendelkezésre álláshoz kapcsolódó Oracle szolgáltatások rövid bemutatása következik. Szó lesz a Real Application Cluster-ről, az adatbázis tükrözésről, a Recovery Manager-ről. Ezek a szolgáltatások képezik a rendelkezésre állás és az adatvédelem központi magját Oracle RDBMS rendszerekben. Olyan megoldások ezek, amelyek használata erősen ajánlott minden komolyabb adatbázissal rendelkező vállalat esetén. Egy informatikai felsővezetőnek nem feltétlenül kell alaposan és részletekbe menően ismernie ezeket a szolgáltatásokat, azonban az mindenképp elvárható, hogy ismerjék az ezekben a megoldásokban rejlő lehetőségeket és az ezekből fakadó üzleti előnyöket. Így lehetőség van a vállalat számára megfelelő koncepció kidolgozására a külső és belső veszélyforrások megelőzésére és az üzletfolytonosság biztosítására.

3.2 Maximum Availability Architecture

A következő fejezetben több Oracle megoldás bemutatása következik, ami a magas rendelkezésre állást és a hibatűrést hivatott biztosítani. Ezek röviden a következőek: Oracle Real Application Cluster, Oracle Data Guard, Oracle Recovery Manager, Oracle Automatic Storage Management. Az Oracle maximális rendelkezésre állást biztosító keretrendszere ezekből az elemekből épül fel, amelyet az alábbi ábra szemléltet.



3. ábra: Oracle Maximum Availability Architecture (Oracle RAC + Oracle Data Guard)

(*Forrás: Oracle, 2010b – 7-19-es fejezet*)

Az MAA előnyei:

- Csökkenti a rendszer implementációs költségeit azáltal, hogy széles körűen konfigurálható és testre szabható.
- Az üzleti igényeknek megfelelően architektúra alakítható ki, amely skálázható teljesítményt biztosít.
- Csökkenthető a tervezett és nem tervezett leállások időtartama.
- Védelmet nyújt a rendszerhibák, alkalmazáshibák, külső támadás és a katasztrófák ellen.
- Alkalmazásával magas szintű SLA elvárások teljesíthetőek.
- Hardver és operációs rendszertől független.
- Kompatibilis egyéb Oracle üzleti megoldásokkal, mint például az Oracle Application Server.

3.2.1. Oracle RAC

Nagy informatikai rendszerek esetén a folyamatos működés, a nagy terhelés, a sok felhasználó, az egyre növekvő tranzakciók száma ma már velejárója az üzletmenetnek. Ezek a tényezők mind jelen vannak egy távközlési cég, egy bank, vagy

akár egy hipermarket adatbázisának működése során, és természetesen ezekre a kihívásokra megoldást is kell nyújtani annak érdekében, hogy az üzletmenet zavartalanul és a legnagyobb biztonságban haladjon. A nagy terhelést el lehet osztani olyan módon, hogy több adatbázis-rendszert építünk ki külön, a lehető legnagyobb teljesítményű szervereken, emellett ugyanúgy a rendelkezésre állás biztosítható tükrözött adatbázisokkal. Ez a megoldás nem feltétlenül a leginkább költséghatékony és optimális, az üzletmenet számára. Többletmunkát jelent az ilyen rendszerek üzemeltetése az adatbázis-adminisztrátorok számára, valamint a folyamatosan növekvő igények miatt a rendszerek bővítése is kérdésessé válhat egy idő után.

Ezekre a kihívásokra választ jelent az Oracle Real Application Cluster megoldása. Az első hasonló törekvések már az Oracle 6.2-es adatbázis-kezelő rendszerében megjelentek, akkor még Oracle Parallel Server néven. A mai elnevezését 2001-ben kapta meg az Oracle 9i megjelenésével. A technológia lényege, hogy a RAC lehetővé teszi, hogy egyetlen adatbázis telepíthetővé váljon több hardverből álló szerverfürtre (clustering), így biztosítva magas rendelkezésre állást, méretrugalmasságot (skálázhatóság), és csökkentett válaszidőt. Így nem szükséges néhány drága szervert vásárolni, elegendő, ha több kisebb teljesítményű hardvert szervezünk fürtökbe. (*Oracle, 2010b*)

Amennyiben Real Application Clustert alkalmazunk, akkor minden egyes adatbázis instanciának hozzáféréssel kell rendelkeznie az Oracle RAC adat-, valamint visszaállítási fájljaihoz. Ennek a feltételnek eleget tenni legegyszerűbben az Oracle ASM-el lehet. Az Automatic Storage Management egy integrált, nagy teljesítményű tároló-kezelő rendszer, amely egyszerűsíti és automatizálja a tárolók kezelését. Segítségével online menedzselhetőek a tárolók, szükség szerint lehet hozzáadni, elvenni, vagy módosítani az adatbázis alá tartozó diszkeket. A kis kitekintés után visszatérve a RAC-ra: szükséges még egy fürtkezelő megoldást is alkalmazni az ASM mellett. Ezt a célt szolgálja az Oracle Clusterware teljes körű, integrált fürtkezelő. Ennek segítségével testre szabhatóvá válik a rendszer és kezelhetővé válnak az esetlegesen felmerülő problémák amellett, hogy nem szükséges 3rd party szoftvereszköz vásárlása. (*Oracle, 2010b*)

Összegzésül elmondható tehát, hogy az Oracle RAC magas rendelkezésre állást biztosít az üzletmenet számára azzal, hogy egy fürt meghibásodása sem befolyásolja

komolyabban a működést, a rendszer továbbra is üzemképes marad. Az alkalmazások terheléselosztása is megoldhatóvá válik, mindegyike külön felügyelhető és szabályozható. A méretrugalmasság érdekében, ha nagyobb feldolgozási kapacitásra van szükség, akkor egy újabb kiszolgálóval történő bővítés egyszerűen megvalósítható. A kulcsszavak tehát: hibatűrés, teljesítménynövekedés, skálázhatóság az alkalmazásokban történő változtatás nélkül. Az ASM-el és az Oracle Clusterware segítségével egy új fogalom született meg: az Oracle Grid Infrastruktúra. Jól illusztrálja ezeknek a megoldásoknak a szerepét és fontosságát a következő idézet Jon Waldron-tól:

> *"High availability is absolutely essential for us… We now use Oracle RAC for instance failover, data guard for site failover, ASM to manage our storage, and Oracle Clusterware to hang the whole thing together."*
>
> **Jon Waldron**
> Executive Architect
> Commonwealth Bank of Australia

Virtualizáció vs. Clustering

Az előzőekben megismerhettük röviden a Real Application Clustering megoldást. Az Oracle Enterprise Grid koncepció emellett támogatja a virtualizációs megoldásokat is, nem zárkózik el az ilyen jellegű technológiai lehetőségektől sem. A virtualizációs megoldások lényege, hogy a nagy teljesítményű szervereken az üzleti igényeknek megfelelően logikai partíciókat hoznak létre, amelyek teljesítménye dinamikusan konfigurálható a rendelkezésre álló erőforrások alapján. Ez többnyire a processzor teljesítmény-, és memória paraméterek skálázhatóságában jelenik meg. Ezáltal az egyes logikai partíciók között a terhelésnek megfelelően lehet elosztani a rendelkezésre álló erőforrásokat, az üzleti folyamatok tapasztalataiból kiindulva. Azt a tényt is érdemes figyelembe venni, hogy egy teszt vagy fejlesztői környezet jóval kevesebb erőforrást igényel, mint egy éles rendszer. Éles rendszer, amely az üzleti folyamatok alapját jelenti, és akár több ezren is használhatják egy időben

párhuzamosan. Az erőforrások racionalizált elosztása az elvárt hatékonyság elérése érdekében fontos tényező, emellett olyan virtualizációból eredő szempontokat is érdemes figyelembe venni, mint például a kevesebb fizikai szerverből adódó alacsonyabb karbantartási költségek, a kevesebb hely-, és energiaigény. A csökkenő leállások számát is ide sorolhatjuk. Az előnyök mellett a hátrányokról is említést kell tenni a szerver virtualizáció esetén: a RAC-cal szemben skálázhatósága limitált, menet közben nem támogatott a szerverbővítés, a szerver beszerzési költségei magasabbak, közepesnél nagyobb rendszerek esetén (pár ezer felhasználó) már kevésbé rugalmas.

A skálázhatóság, az elérhetőség, a flexibilitás és az előnyök különböző szempontokból történő összehasonlítását az alábbi táblázatban láthatjuk összefoglalva:

Virtualizáció	Klaszterezési megoldások
▪ A szerver fizikai adottságai (memória, processzor) korlátokat szabnak. ▪ A logikai partíciók között lehetséges erőforrás allokáció. ▪ Nagyobb teljesítményt új szerverre történő migrációval lehet elérni.	▪ A fürt akár korlátlan mértékben növelhető horizontálisan. ▪ Szerverek kiesése és a fürt bővítése nem jelent problémát akár az üzleti folyamatok közben sem. ▪ Teljesítmény növelése a megfelelő erőforrás allokációval menet közben elvégezhető.
▪ A tervezett leállások idejét csökkenti, megfelelő megoldások alkalmazásával magas rendelkezésre állást biztosít.	▪ A tervezett és nem tervezett leállások idejét is nagymértékben csökkenti. Magas rendelkezésre állást biztosít.
▪ Jól menedzselhető szerver szinten az erőforrás kiosztás az egyes logikai partíciók között	▪ Szerver léptékű erőforrás kiosztás. Dinamikus terhelés elosztás.
▪ Megnövelt erőforrás kihasználás. Kevesebb karbantartási feladat. Egyszerűbb kialakítás	▪ Kisebb teljesítményű, olcsóbb szerverekből megnövekedett teljesítmény és rendelkezésre állás.
▪ Kevésbé kritikus rendszerek esetében célszerű alkalmazni, fejlesztői és tesztelői környezeteken. ▪ Kicsitől a közepes rendszerekig	▪ Közepestől a nagy rendszerekig. Üzletileg kritikus rendszerek esetén. Komplexebb telepítést és karbantartást igényel.

2. táblázat: Virtualizáció vs Clustering (*Forrás: Oracle, 2010b*)

3.2.2 Data Guard

Az előzőekben az Oracle Real Application Cluster-t ismerhettük meg nagyon röviden. A RAC biztosítja egy adott lokális helyszínen lévő fürtök magas rendelkezésre állását. Azonban ha ezt a helyszínt katasztrófa éri, akkor a RAC sem mentheti meg önmagában a vállalat adatait. Az adatbázisok tükrözése (mirroring) jelentheti erre a problémára a megoldást. Az Oracle Data Guard szolgáltatása hivatott ezt a funkciót

betölteni olyan módon, hogy az éles rendszerek mellett bármikor használatra kész standby rendszereket biztosít. A Real Application Cluster kiegészítve a Data Guard-dal együtt jelenti az Oracle által „best practice”-ként ajánlott Oracle Maximum Availability Architecture-t, amely a lehető legmagasabb rendelkezésre állást és adatvédelmet biztosítja.

Az Oracle Data Guard egy olyan szoftvermegoldás, ami biztosítja egy vagy több standby adatbázis létrehozásával, felügyeletével, karbantartásával járó feladatokat, ezzel védelmet nyújt a rendszerhibáktól, katasztrófahelyzetektől, adathibáktól. Lehetővé válik a vállalat számára, hogy két vagy több földrajzilag elkülönülő helyen - például két szerverparkban - helyezze el adatbázis-szervereit. Az egyik helyen működtetheti éles rendszereit a másik helyen pedig a Data Guard által biztosítva konzisztensen, naprakészen, folyamatosan frissülve a standby adatbázisok találhatók. A két helyszín közötti távolságra megkötés nincs, egyedül a kommunikációt kell biztosítani. Az első hasonló megoldások már a 7.3-as (1996) verziójú Oracle adatbázis-kezelőben megjelentek, a Data Guard azonban csak a 9i verziótól elérhető. (*Oracle, 2010c*)

A standby adatbázis az éles mentéséből készül. Ezt folyamatosan frissíti a Data Guard szolgáltatás, biztosítva így a két adatbázis közötti konzisztenciát. A frissítés feltétele, hogy az éles, elsődleges (primary) adatbázis `ARCHIVELOG` módban legyen, hogy a változások `archived (redo) log` fájlokban megjelenjenek, és ezeket továbbítsa a Data Guard a standby, azaz másodlagos oldal felé. Ezek az `archived log` fájlok tartalmaznak minden változást, amelyek az adatfájlokat érintik, ezek segítségével valósítható meg a standby adatbázis-környezet. A Data Guard-nak 3 különböző adatvédelmi üzemmódját különböztethetjük meg aszerint, hogy a vállalat hogyan értékeli a teljesítményt, az adatvesztés kockázatát és a rendelkezésre állást. Ez 3 üzemmód a következő:

- *Maximális védelem*

  Ez a mód biztosítja a legmagasabb szintű adatvédelmet: a primary adatbázis leállása esetén sem merülhet fel adatveszteség. A tranzakciók során keletkező logfájlok rögtön továbbítódnak a standby oldal felé. A primary adatbázisban a tranzakciók addig nem véglegesítődnek (nem történik commit), amíg a megfelelő logfájlok legalább az egyik meghatározott célhelyen rendelkezésre

nem állnak. Az adatvesztést megelőző lépés ebben a védelmi módban az, hogy a primary adatbázis leáll, ha nem képes a log fájlokat legalább egy standby adatbázissal szinkronizálni. Ezért érdemes minimum 2 standby adatbázissal számolni maximális védelem esetén.

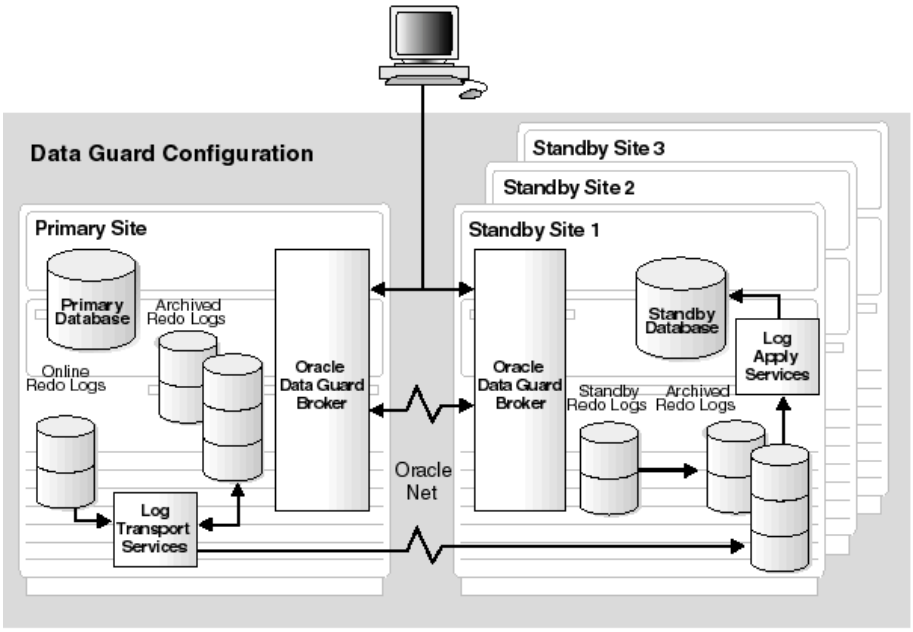
- *Maximális rendelkezésre állás*

Ez a mód biztosítja a legmagasabb adatvédelmet amellett, hogy ne veszélyeztesse a primary adatbázis elérhetőségét. A változások nem véglegesítődnek ebben az esetben sem, amíg nem sikerül az `archived log` fájlokat szinkronizálni a standby adatbázissal. Amennyiben a primary adatbázis nem képes a log fájlokat eljuttatni a standby oldalra, akkor a maximális teljesítmény módban megfogalmazottak alapján jár el.

- *Maximális teljesítmény*

Ez a védelmi mód biztosítja a lehető legmagasabb adatvédelmet, figyelembe véve azt, hogy ne veszélyeztesse a primary adatbázis teljesítményét. A változások azonnal véglegesítődnek, amint a redo log fájlba bekerülnek. A logfájlok csak ezután kerülnek továbbításra a standby oldal felé. Ez az alapértelmezett védelmi mód az Oracle adatbázisok esetén. Az adatvédelemben elmarad a másik két módszerhez képest, azonban hatása a primary adatbázis teljesítményében minimális, ellentétben a maximális rendelkezésre állással és maximális védelemmel szemben.

(*Forrás: Oracle, 2010c*)



4. ábra: Oracle Data Guard
(*Forrás: Oracle, 2010c, 1-4 fejezet*)

A standby adatbázisoknak is megkülönböztetjük két alapvető fajtáját. A fizikai standby adatbázis, egy a primary adatbázissal tökéletesen megegyező adatbázis, ezalatt blokkról blokkra azonos adatbázis-struktúrát értünk. Minden séma és minden index megegyezik a két adatbázisban. Ezekhez a standby adatbázisokhoz az Oracle a *Redo Apply* módszert alkalmazza, amely azt jelenti, hogy a standby oldal felé továbbított `archived redo log` fájlok segítségével tudja szinkronban tartani a két adatbázist. A logikai standby adatbázisok a primary adatbázissal logikai szinten azonos információkat tartalmaznak, azonban az adatok struktúrája és szervezése eltérő lehet. A szinkronizáció során az Oracle az *SQL Apply* módszert használja, amelynek lényege, hogy a primary adatbázis redo log fájljait SQL utasításokká alakítja át, majd ezeket alkalmazza és hajtja végre a standby oldalon. Ezzel lehetővé teszi a felhasználók számára az Oracle, hogy az utasítások végrehajtása mellett párhuzamosan lekérdezéseket futtassanak a standby adatbázison. Így a standby adatbázis kétféle célt is szolgál egyidejűleg: adatvédelmet, valamint kimutatás készítést is. (Kevin Loney, 2006)

Az Data Guard az eddig bemutatottakon kívül az éles és standby adatbázisok közötti tervezett átkapcsolást (switchover) és a probléma esetén felmerülő átkapcsolás (failover) menedzselésére, ezáltal biztosítva azt, hogy a lehető legkevesebb időt álljon az adatbázis. Ezek a lehetőségek jól konfigurált rendszer esetén mindösszesen egy

utasítás kiadásával megvalósíthatók. Ez azonban még mindig nem minden. A Data Guard szolgáltatás támogatja az úgynevezett witness server alkalmazását is. Ez egy olyan szolgáltatás, amelynek keretében az elsődleges adatbázistól eltérő helyen egy független szerver monitorozza az éles és standby adatbázisok állapotát és kapcsolatát. Szükséges esetben pedig automatikusan közbeavatkozik és végrehajtja akár a failover mechanizmusát is. A witness server monitorozhatja az éles adatbázis elérhetőségét, a standby szinkronitását, az archived log fájlok átvitelét, és akár riasztást is küldhet, ha a standby adatbázis nincs szinkronban kellőképpen az éles rendszerrel. A rendszer személyre szabottan konfigurálható: például, ha az elsődleges adatbázis 5 percen keresztül nem elérhető, akkor automatikusan végrehajtódhat a failover a standby adatbázisra.

3.3. RMAN

Minden éles környezetben működő adatbázis esetében szükség van a lehetséges hardverhibák, helytelen működés, külső támadások vagy akár felhasználói tevékenységből fakadó adatvesztés, adat-meghibásodás, adatkorrupció kezelésére. Hosszasan sorolhatnánk a különböző veszélyforrásokat, amelyek olyan hatással lehetnek a szervezet működésére, hogy megelőző lépéseket kell biztosítani a folyamatos üzletmenet érdekében. Ezeket a lépéseket támogatja többek között az RMAN is.

A Recovery Manager. Az Oracle adatbázis-kezelő rendszer történetét tekintve 1997-ben jelent meg az Oracle eszközei között a 8-as verziójú adatbázis-kezelő rendszer részeként, ez azt jelenti, hogy külön telepítést sem igényel a felhasználó részéről. Röviden összefoglalva ez az eszköz hivatott megvalósítani az adatbázis mentési folyamatait, a helyreállítási és visszaállítási funkcióit. Az elmúlt több mint egy évtized alatt azonban funkcionalitása és szerepe jelentősen bővült, így biztosítva egy rendkívül hasznos eszközkészletet az adatbázis adminisztrátorok számára. Ilyen funkciók lehetnek példaként említve a teljes adatbázis, a táblaterek, adatfájlok, control fájlok mentése és visszaállítása; a céladatbázis duplikálása, és akár adott időpontra történő visszaállítása archive log fájlok alapján. Kezelése saját utasításkészlettel – ami nagyban hasonlít az Oracle adatbázisok utasításkészletére – történik, ami által könnyen

és formalizáltan megvalósíthatók, és akár ütemezhetők is a backup and recovery feladatok. Amint tehát láthatjuk, a Recovery Manager számos hasznos és nélkülözhetetlen funkcióval rendelkezik, amelyeket a hagyományos export-import (ma már expdp és impdp) funkcióval nem lehet megvalósítani. Többen tekintenek ugyanis az export-import funkcióra, mint adatbázis mentési funkcióra, azonban ezt nem szabad adatbázis mentési folyamatnak tekinteni. Korántsem biztos, és semmiképp sem garantált az, hogy egy teljes adatbázis exportot sikeresen vissza lehet importálni adott szituációban.

Az RMAN környezet több összetevőből épül fel. Mindez az alábbi elemeket tartalmazza:

- Céladatbázis (target database)	- Recovery katalógus (recovery catalog)
- RMAN kliens (RMAN client)	- Recovery katalógus séma felhasználó
- Flash recovery area	- Media manager

3. táblázat: Az RMAN környezet összetevői (*Forrás: Oracle, 2010e*)

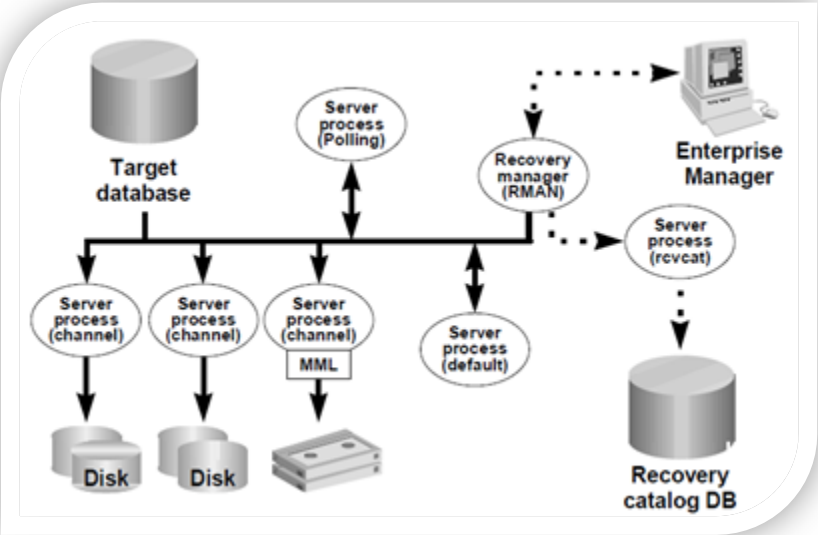
A *céladatbázis* az éles adatbázis rendszer, amelyet menteni szeretnénk. Ez az az adatbázis, amely tartalmazza a control fájlokat, az adatfájlokat, archived log fájlokat, amelyeket az RMAN-nal menteni, és szükség esetén visszatölteni kívánunk. A mentés és visszaállítás ezt az adatbázist érinti. Az *RMAN kliens* egy alkalmazás, amely vezérli és felügyeli a mentést és visszaállítást a céladatbázison. A céladatbázis eléréséhez az Oracle Net-et használja, így tulajdonképpen bármelyik host-ról elérhető a céladatbázis ezen a szolgáltatáson keresztül. A *flash recovery area* egy meghatározott hely a lemezen ahova a mentéssel kapcsolatos fájlok kerülnek, mint például az adatbázisról készült mentés, a redo logok, vagy pedig az archive logok. Ez a hely általában az adatbázis inicializáló paraméterei között kerül definiálásra (`db_recovery_file_dest`). A *recovery katalógus* szolgál mintegy repository-ként egy-vagy több adatbázis metaadatainak tárolására. Javasolt azonban, hogy minden target adatbázis számára saját katalógust hozzunk létre. Amennyiben a control fájlok megsérülnek, vagy elvesznek, a katalógus segítségével – ha nem is az eredeti állapotában de – könnyedén visszaállítható a szükséges fájl. Egy céladatbázisnál is, de többnél mindenképp ajánlott egy külön adatbázist létrehozni az *RMAN séma felhasználó* és a recovery katalógus

számára. Végezetül pedig a *media manager* biztosítja azokat a szolgáltatásokat és funkciókat, amelyek segítségével akár külön device-ra vagy szalagra lehet a mentéseket végezni, használata nem kötelező. (*Oracle, 2010e*)

Adatbázis-mentések szempontjából megkülönböztetünk kétfajta módszert, amelyeket természetesen az RMAN is támogat. Ez a két módszer a cold (offline vagy konzisztens) és a hot (online vagy inkonzisztens) backup. A cold mentések során tipikusan a teljes adatbázisról készül mentés. Ekkor az adatbázist le kell állítani, hogy a mentés során az adatfájlok konzisztenciája megmaradjon, így egy konkrét időpontbeli állapotot tükröz a mentés. Ez a módszer azt a veszélyt rejti magában, hogy ha nincs az adatbázis `ARCHIVELOG` módban, akkor a mentés és helyreállítás között eltelt időtartam alatt történő változások – legyenek azok akár adatszerkezeti vagy adattartalombeli változások – elvesznek, tehát adatveszteséggel kell számolni. Ezt a veszélyt nem engedheti meg egy komolyabb adatbázissal rendelkező szervezet sem, így ezt a mentési módot nem célszerű választani éles, sok tranzakciót kezelő adatbázison, amelyik túlnyomó többségben 24x7-es környezetben működik. Viszont előre meghatározott időszakonként, adott időpontot tükröző mentéseket célszerű lehet ily módon elvégezni az éles adatbázison is, továbbá egyéb adatbázisokon – tipikusan fejlesztői adatbázisokon a projektek előrehaladtával, az egyes mérföldkövek után – szintén érdemes lehet ezt a mentési módszert alkalmazni. Így bármikor egy adott időpontot tükröző állapot visszaállíthatóvá válik.

Ennek a mentési módnak a hátrányait küszöböli ki a hot vagy pedig más néven online adatbázismentés. Az online mentés során lehetőség van csak néhány állomány mentésére is, pl. adatfájlok mentésére is. Ennek a mentési módnak feltétele, hogy az adatbázis `ARCHIVELOG` módban legyen, amelynek során folyamatosan keletkeznek `archive log` fájlok a `redo log` fájlok megtelésével. (Az adatbázis a tranzakció során keletkező változásokat folyamatosan írja a redo log fájlokba, ezért a `LGWR` háttérfolyamat a felelős. Ha az egyik fájl megtelik, akkor elkezdi a következőt írni, azonban ha mindegyik redo log fájl eléri a maximális méretét, akkor a fájlok tartalma törlődik és kezdődik a folyamat az elejétől. Az `ARCHIVELOG` mód során, ha egy redo logfájl megtelik, akkor annak tartalma automatikusan `archived log` fájl formájában a megadott helyre mentődik, az `ARCH` háttérfolyamat segítségével.) Az adatbázisról menet közben készül mentés anélkül, hogy leállítanák, így nem történik kiesés az

üzletmenetben. Az adatbázis visszatöltésekor – legyen az akár egy specifikált időpontra vagy a legutolsó elérhető állapotra történő visszatöltés – az archive log fájlok biztosítják, hogy ne történjen adatvesztés a folyamat során. Ezért tehát célszerű a log fájlok legalább két helyen történő tárolása a `log_archive_dest` paraméterek beállításának segítségével. Célszerű úgy optimalizálni a mentéseket, hogy lehetőleg kevés DML-tranzakció során történjen meg a mentési folyamat. (*Kevin Loney, 2006*)



5. ábra: RMAN arhitektúra

(*Forrás: Oracle, 2010d*)

A Recovery Manager, mint láthattuk, a szolgáltatások széles körét nyújtja, ha mentési, helyreállítási feladatokról van szó. Összefoglalva, előnyei a következők:

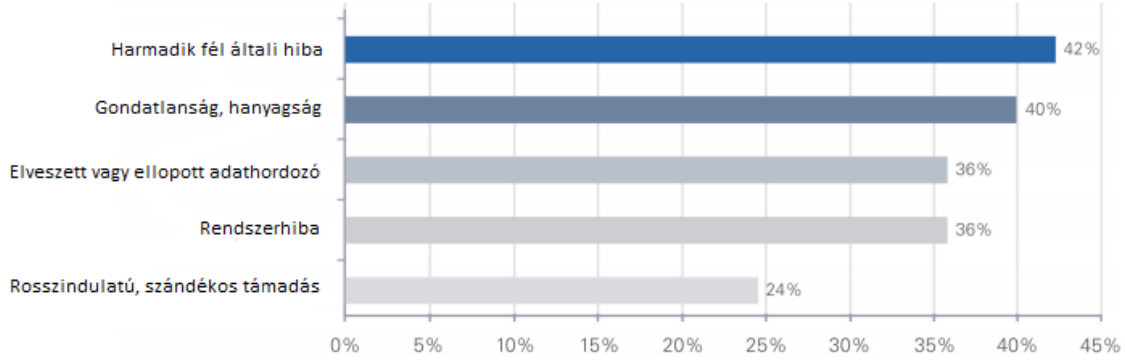
- Több adatbázis metaadatait tárolhatjuk egyetlen katalógusban
- Könnyen és gyorsan szinkronizálhatók az adatok a céladatbázis és a metaadatbázis között
- Scripteket tárolhatunk a recovery catalog-ban
- A mentések széles köre és formája támogatott
- Könnyen személyre szabható és automatizálható

A hátrányok között meg kell említeni, hogy a katalógus karbantartása, menedzselése többletmunkát jelent az adatbázis-adminisztrátorok számára. Ezen kívül ki kell alakítani hozzá egy saját, külön erre a célra szolgáló adatbázist. Ennek természetesen tárhellyel

kapcsolatos követelményei is vannak, rendelkezésre kell bocsátani a szükséges szabad helyet. Természetesen a metaadatokról is célszerű mentéseket készíteni, valamint a katalógust is, mint szoftvert, az idő folyamán frissíteni szükséges. (*Oracle, 2010d*)

3.4. Audit - a felhasználók nyomon követése

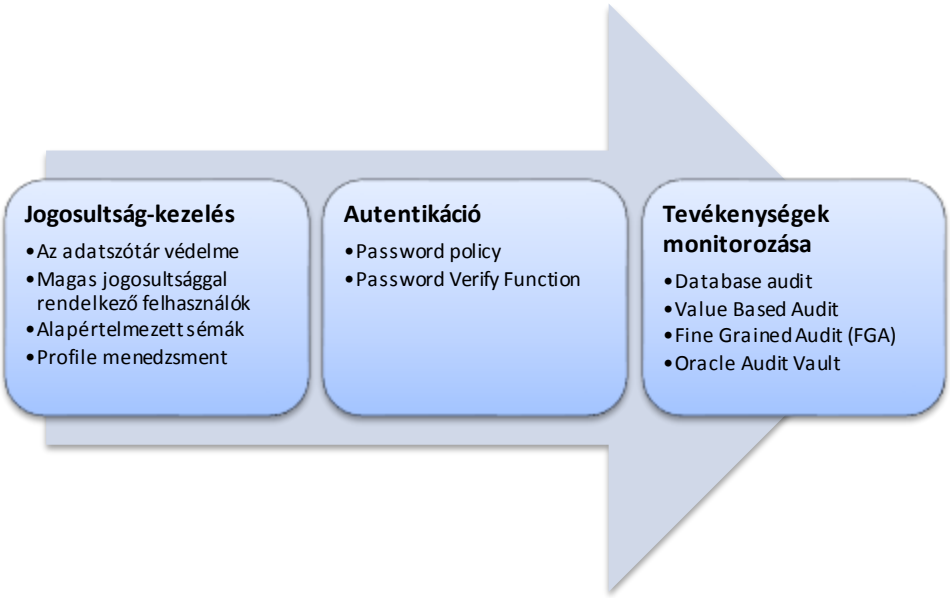
Az adatbázisok védelme a külső támadásoktól, hardveres és szoftveres meghibásodásoktól, előre nem várt helyzetektől elengedhetetlenül fontos. De ez azonban nem minden. Az adatbázis tükrözése vagy klaszterezése segíthet sok problémát megelőzni, de ezek a megoldások sem alkalmasak a vállalaton belülről jelentkező fenyegetések maximális kezelésére. Ezekre más megoldások szükségesek. Szükségesek, mert egyre nagyobb méreteket ölt az adatszivárgások, adatvesztések mértéke. A Penomenon Institute 2010-es felmérése is ezt támasztja alá. Ebben a felmérésben részletesen foglalkoznak az adatszivárgások okaival, költségeivel és mindezek tendenciáival. Az adatszivárgások különböző okainak gyakoriságát az alábbi ábra mutatja a felmérés alapján:



6. ábra: Az adatszivárgás okai (*Forrás: Penomenon Institute, 2010 alapján*)

Többféle nézőpontból megközelíthetjük az adataink, az adatbázisok védelmét. Korlátozhatjuk az adatokhoz, szolgáltatásokhoz, alkalmazásokhoz való hozzáférést, megfelelő autentikációs folyamatot biztosíthatunk, vagy pedig felügyelhetjük, monitorozhatjuk a gyanús aktivitásokat. A felhasználók nem férhetnek hozzá minden adathoz az adatbázisban: az adatok jellegétől függően lehetnek személyes adatok, üzleti titoknak minősülő információk, és számos egyéb bizalmas információ. Alapvetően a koncepcionális elképzelés az, hogy a felhasználók csak azokhoz az adatokhoz férjenek hozzá, amelyek munkavégzésükhöz feltétlen szükségesek. Az

Oracle adatbázis-kezelő rendszer természetesen képes ezeket a problémákat orvosolni: az auditálási megoldások sokrétűsége, a felhasználók jogainak széles körű testreszabhatósága, a tevékenységeik nyomon követése biztosított a rendszer által, különböző megoldások keretében.



4. táblázat: Biztonsági megoldások

Három alapvető megközelítési irányvonalat mutat a fenti ábra, néhány példával illusztrálva. Ezeket az irányvonalakat az Oracle adatbázis-kezelő rendszer képes lefedni, minden igényt kiszolgálva. Széles körű testre szabhatóságot, rugalmas megoldásokat képes nyújtani az adatbázis-adminisztrátorok számára, ezáltal biztosítva az üzletfolytonosság-menedzsmentben foglalt kritériumok teljesítését. Az első kérdéskör a jogosultságkezelési megoldások. Az Oracle adatbázis-kezelő eszközöket biztosít, amelyekkel megvalósítható az, hogy mindenki csak azokhoz az objektumokhoz férhessen hozzá, amikhez feltétlenül szükséges. Ezeket a korlátozásokat a különböző szerepkörökön (`role`) keresztül lehet megvalósítani. A számos alapértelmezett szerepkörön (`connect`, `resource`, `dba`) kívül saját testreszabott szerepkörök definiálhatóak az alkalmazásoktól, adatbázis objektumoktól függően és egyéb Oracle funkcióktól függően. Mindez alatt a különböző jogosultságok (`SELECT`, `UPDATE`, `DELETE`, `ALTER`, `EXECUTE`, `RECOVERY_CATALOG_OWNER`, stb.) értendőek. Így adatbázis-környezetünktől függően egy egyedi koncepció valósítható meg, saját szerepkörök által, amelyek leképezik az üzleti igényeket és megfelelnek a biztonsági előírásoknak. A jogosultság kérdéséhez közvetetten kapcsolódik a felhasználói profil

beállításainak lehetősége az adatbázis-kezelő rendszerben. Ennek keretében az adatbázis biztonságos működésének érdekében korlátozások állíthatók be a felhasználói és üzleti/technikai sémákra. Ezek a maximális egyidejű bejelentkezések számát, a processzorhasználatot érintő paramétereket, a csatlakozási és inaktivitási időtartamot, valamint az SQL utasítások erőforrásigényét is befolyásolják. Ezen kívül korlátozható a felhasználó által használt tárhely terület nagysága is. Mindezek a lehetőségek igen fontos szereppel bírnak, ezektől eltekinteni nem lehetséges. (*Natan, 2009*)

A második lépcsőfokot az autentikáció kérdésköre alkotja. A megfelelő jogosultságok beállítása után a felhasználók használhatják az adatbázis-kezelő rendszer által nyújtott szolgáltatásokat, természetesen a konfiguráció után ki kisebb, ki nagyobb jogkörrel. A `sysdba` és `dba` szerepkörök igen széles jogkörrel bírnak, szinte bármilyen objektumot törölhetnek, módosíthatnak az adatbázisban. Ezekkel a role-okkal rendelkező felhasználókat, sémákat olyan jelszavakkal kell ellátni, amelyek nehezen feltörhetőek, tehát sok karakterből állnak, kis és nagy betűk mellett speciális karaktereket is tartalmaznak. De emellett magától értetődően a hétköznapi felhasználói jelszavaknak is egy koncepciót kell kialakítani, ami megfelel a biztonsági előírásoknak. Az Oracle adatbázis-kezelő a beépített funkciók (password verify function) segítségével képes segítséget nyújtani annak érdekében, hogy megfelelő jelszókezelési módszerek kerüljenek bevezetésre.

Végezetül, ha kialakításra került a megfelelő jogosultság kezelési szisztéma és az autentikációs koncepció, akkor az érzékeny adatok védelme és az esetleges előírások érdekében az auditálás marad hátra. Nyomon követni a felhasználói tevékenységet, monitorozni a magas privilégiumokkal rendelkező user-eket és ezekből információkat kinyerni minden vállalat érdeke. Az auditálás célja lehet a nem szokványos tevékenységek felderítése, a hozzáférési jogok megkerülésének vizsgálata is. Az auditjelentések sok hasznos információval szolgálhatnak. Azonban, hogy ezek valóban hasznosak legyenek, nem elég az adatokat gyűjteni és riportokat előállítani belőlük, hanem azokat ténylegesen fel kell dolgozni, és többletinformációt nyerni belőlük. Az olyan auditjelentésnek, adatnak, amelyet senki nem használ fel, nem sok érdemi haszna van. Az Oracle adatbázis-kezelő rendszert alapul véve többfajta megoldást alkalmazhatunk külön-külön, de akár egyszerre is. A legegyszerűbb

megoldások az egyéni fejlesztés által létrejövő mechanizmusok, amelyek audit információkat hoznak létre. Ilyen megoldások lehetnek a triggerek, PL/SQL utasítások használata. A beépített megoldások között érdemes használni az általános adatbázis auditot, mint szolgáltatást, amelyet az Oracle nyújt. Ennek keretében meghatározható, hogy az adatbázis-kezelő milyen esetekben tároljon el automatikusan információkat. Ezek lehetnek egyszerű utasítások (select, update), vagy akár események is, mint például az adatbázishoz csatlakozás. A legjobban testre szabható megoldás a Fine Grained Audit, amely által egyénileg definiálható szabályok hozhatók létre. Az itt leírtak a gyakorlati résznél nyernek teljes értelmet, amikor példákon keresztül bemutatva láthatóvá válik az egyes megoldások közötti különbség. Mindezek az audit megoldások, amelyet az Oracle adatbázis-kezelő rendszer nyújt hivatottak megelőzni az esetleges felhasználói tevékenységből fakadó veszélyeket. A legtöbb vállalat esetében ezeknek a lehetőségeknek az alkalmazása elegendő biztonságot kell, hogy nyújtson az üzletfolytonosság menedzsment irányelveknek megfelelően.

3.5. Mit érdemes figyelembe venni?

Ebben az alfejezetben néhány egyéb nézőpontot kívánok megvilágítani, amelyeket érdemes figyelembe venni az informatikai rendszerek vizsgálata során.

3.5.1. Tervezett és nem tervezett leállások

A legnagyobb kihívás az informatikai rendszerek üzemeltetése esetén, hogy a rendszer elérhetősége 100%-os legyen a nap minden órájában és az év minden napján. Azonban a leállások minden informatikai rendszer esetén elkerülhetetlen velejárója az üzletmenetnek. Alkalmazhatunk bármilyen megoldást ezt kiküszöbölni nem lehet. Az okok között szerepelhet számtalan indok: hardverek cseréje, fejlesztése, karbantartása, a szoftverek frissítése, az alkalmazások migrációja. És eddig még csak a szükséges leállásokról beszéltem, amelyek előre tervezhetőek, amelyek mellett a nem várt leállások is nehezíthetik a folytonos üzletmenetet. Ide sorolhatók a külső forrásból származó veszélyek, mint például az áramkimaradás, a hardverhibák, az alkalmazáshibák, de a felhasználók tevékenységéből fakadó belső veszélyforrások is. Minél összetettebb egy rendszer, minél több alkalmazás kapcsolódik egymáshoz, annál

nagyobb kihívást jelent ez az üzemeltetés számára. A hibalehetőségek hatványozottan növekednek Mindezek alapján a leállások kétféle csoportba sorolhatóak: az előre tervezett, jól menedzselhető, valamint a nem várt, nagy veszélyeket magában foglaló nem tervezett leállások. Ezeknek a leállásoknak a hatásait hivatott az üzletmenet-folytonosság minimalizálni. (*Raffai, 1999*)

A korábban bemutatott Oracle megoldások, amik az adatbázisok biztonságát, és magas rendelkezésre állását biztosítják, szintén csoportosíthatók, aszerint, hogy melyik típusú leállás ellen nyújtanak védelmet (*Oracle, 2010b*).

Tervezett leállások esetén:

- Karbantartás, fejlesztés (szoftver és hardver is)
    - Standby adatbázis (Data Guard)
    - Automatic Storage Management

Nem tervezett leállások esetén:

- Külső veszélyforrás
    - Real Application Cluster
    - Standby adatbázis (Data Guard)
- Hardverhiba
    - Real Application Cluster
    - Standby adatbázis (Data Guard)
    - Automatic Storage Management
- Alkalmazáshiba
    - Recovery Manager
    - Real Application Testing
- Adathiba
    - Recovery Manager
    - Automatic Storage Management
    - Oracle Flashback Technology
- Belső veszélyforrás
    - Audit Vault
    - Total Recall
    - Standard database audit
    - Profile management

3.5.2. Funkcionális környezetek védelme

Minden vállalat esetén szükség van több olyan környezetekre az éles rendszerek mellett ahol az egyes fejlesztések, tesztelések és egyéb ehhez kapcsolódó tevékenységek elvégezhetők. Alapvető elvárás az, hogy nem az éles rendszeren történik a tesztelés sem. Egy nagyvállalat esetén mindez akár 3-4 környezetet jelenthet az éles környezeten kívül. Beszélhetünk fejlesztői-, teszt-, integrációs környezetekről is, amelyeknek jól specializált célja van. A fejlesztéseknek, rendszerváltozásoknak minden lépcsőfokon keresztül kell mennie annak érdekében, hogy az éles rendszerbe is bekerüljenek. Felmerülhet a kérdés, milyen módszereket kell alkalmazni ezeknek a környezetek rendelkezésre állásának biztosítására. Az előzőekben felsorolt Oracle megoldások közül kell kiválasztani azokat, amelyek feltétlen szükségesek ezeknek a rendszereknek a védelme érdekében. Természetesen ezeknél az adatbázisoknál nem szükséges olyan mértékű rendelkezésre állás biztosítása, mint az éles rendszereknél. Nem jelent az sem súlyos problémát, ha a rendszerek csak néhány óra elteltével válnak használhatóvá egyes leállások után. Ennek következtében az adatbázis tükrözés ezeknél a rendszereknél nem szükséges, mint ahogy a klaszterezés sem elvárás. Ezeknél az adatbázisoknál a hibák még megengedettek - sőt, céljuk a hibák felderítése még az éles rendszerbe kerülés előtt - ezért sok visszatöltés jellemzi ezeket a környezeteket. Ennek menedzselésére az RMAN használható az általános export-import funkció mellett. Ezeknél a környezeteknél egy újfajta problémát kell viszont mindenképp kezelni: az éles rendszerekből származó érzékeny adatok titkosítása az illetéktelek elől. Ez a deperszonalizáció. A teszt-, és fejlesztői adatbázisok természetesen az éles adatbázisból készülnek, és azok adatait tartalmazzák. Ha a vállalat külső fejlesztőcégeket alkalmaz, akkor az üzleti adatok titkosítása feltétlen szükséges. Ennek többféle megoldása lehetséges: az adatok összekeverhetők, torzíthatók vagy akár törölhetőek is. Figyelni kell azonban arra, hogy az adatok közötti összefüggések és kapcsolatok ne sérüljenek.

A deperszonalizációnak két módja lehetséges: saját, egyedi megoldás fejlesztése, vagy külső kész megoldás alkalmazása. A saját fejlesztés előnye, hogy olcsó és belső fejlesztés keretében megvalósítható. Hátránya ebből fakadóan, hogy megvalósítása jóval több ideig elhúzódhat. A külső megoldás alkalmazása és annak konfigurálása jóval

rövidebb idő alatt megvalósítható, de költségesebb is, mint az egyedi fejlesztés. A külső megoldások között érdemes megemlíteni az Oracle Data Masking Pack (ODMP) megoldást, amely teljes mértékben Oracle adatbázisokra optimalizált. Az ODMP képes az érzékeny éles adatokat felkutatni és irreverzibilis módon fiktív adatokká konvertálni azokat. Az adatmaszkolás során számos egyéb körülményt is képes figyelembe venni. Az ODMP használatával biztonságos, és nagy teljesítményű adatmaszkolás valósítható meg gyorsan és könnyedén.

3.5.3. OLTP és Data Warehouse

A korábbi fejezetekben megismerkedhettünk az adatbázis-rendszerek fogalmával, különböző típusaival és azok tulajdonságaival. Az egyes típusok tulajdonságai, funkcionalitása alapvetően meghatározza az azokkal szembeni elvárásokat, így ide értve a rendelkezésre állással, hibatűréssel kapcsolatos tényezőket is. Mindezek figyelembevételével, az előzetes igények felmérésével alakítható ki megfelelő védelem adatbázisaink számára. A tranzakció feldolgozásra szánt adatbázisok, és az adattárházak – esetleg archív adatbázisok – ezek alapján mind egyedi védelmi megoldásokat igényelnek.

A tranzakció-feldolgozó adatbázisokat több száz, de akár több ezer felhasználó is használhatja egyszerre, ezért a rendelkezésre állása az üzletmenet szempontjából kritikus. Többnyire rövidebb és gyakoribb tranzakciók hajtódnak végre, kevesebb DML művelettel, amelyek kevés elemet is érintenek. Adatforrása homogén, és mindig az aktuális állapotokat mutatják az adatok az adatbázisban. Méretét tekintve kisebb adatbázisokról van szó, mint az adattárházak esetén. Egy ilyen adatbázis esetén a terheléselosztás érdekében indokoltak lehetnek a szerver-klaszterezési megoldások. Amennyiben ezt a lehetőséget nem alkalmazzák, akkor elengedhetetlen az adatbázis replikálása és legalább egy folyamatosan frissülő, de akár több standby adatbázis környezet kialakítása, a jelentkező problémák kiküszöbölésének érdekében. Ezeket a standby adatbázisokat azonban lehet más célokra is használni: mentések készíthetők ezek alapján, lekérdezéseket, elemzéseket lehet végezni rajtuk, így megtakarítva értékes erőforrásokat az éles primary rendszereken. Emellett időközönként ajánlott teljes adatbázismentéseket végezni, valamint az archived logok mentése is javallott,

abban az esetben, ha egy adott időpontot tükröző állapotra szeretnénk visszaállítani, vagy rosszabb esetben helyreállítani az adatbázist.

Az adattárházak (data warehouse) esetén kissé más a helyzet, ennek megfelelően eltérő megoldásokat is lehet alkalmazni. Az adattárházak adattartama historikus, és több különböző forrásból származik. Továbbá jellegzetessége a ritkább, de hosszabb ideig tartó tranzakciók, amelyek több eltérő adatelemet is érintenek. A hangsúly elsődlegesen az elemzési feladatokon van. Jóval több adatot tartalmaznak a tranzakció-feldolgozó rendszerekhez képest (Online Transaction Processing - OLTP), mivel általában rendszeres időközönként az áttöltő folyamatok keretében az egyes társrendszerekből ide kerülnek át az adatok, ezért egynél több standby adatbázis kialakítása jelentősebb erőforrásokat leköthet, főként, ha csupán lemezterületre gondolunk. Emellett viszont kevesebb felhasználó van jelen a rendszerben egy időben, a hangsúly az adatok feldolgozásán és az elemzéseken van. Az adattárházak esetén is megvalósítható a szerver-klaszterezés, csak ebben az esetben viszonylag állandó teljesítményi elvárásoknak kell megfelelni, így egy jól konfigurált környezeten ritkán kell változásokat eszközölni. Az adatbázis backupok itt is fontosak, azonban ritkábban esedékesek, mint egy OLTP rendszernél, főképp abból kifolyólag, hogy az adattárházakból a legtöbb esetben már csak az archív adatbázisba kerülnek adatok.

OLTP és adattárház megoldásoknál sem tekinthetünk el attól a ténytől, hogy a felhasználók megfelelő menedzselése kritikus fontosságú, az üzletfolytonosság és az adatok védelme érdekében, de ez általánosságban az adatbázis-rendszerek esetén is elmondható. Ezzel megelőzhetünk sok kritikus problémát is, tehát az adatbázisok konfigurálása ebből a szempontból mindenképp javallott. A jogosultságkezelés, a kvótabeállítások, az adatbázis-objektumok kezelése, a profilbeállítások mellett érdemes figyelmet fordítani a felhasználók és az üzleti folyamatok auditálására is. Amint tehát láthatjuk ez egy összetett témakör, amely átfogóan érinti az adatbázis-kezelő rendszereket. Ennek kezelése – függetlenül attól, hogy OLTP vagy adattárház rendszerről beszélünk – mindenképpen kritikus.

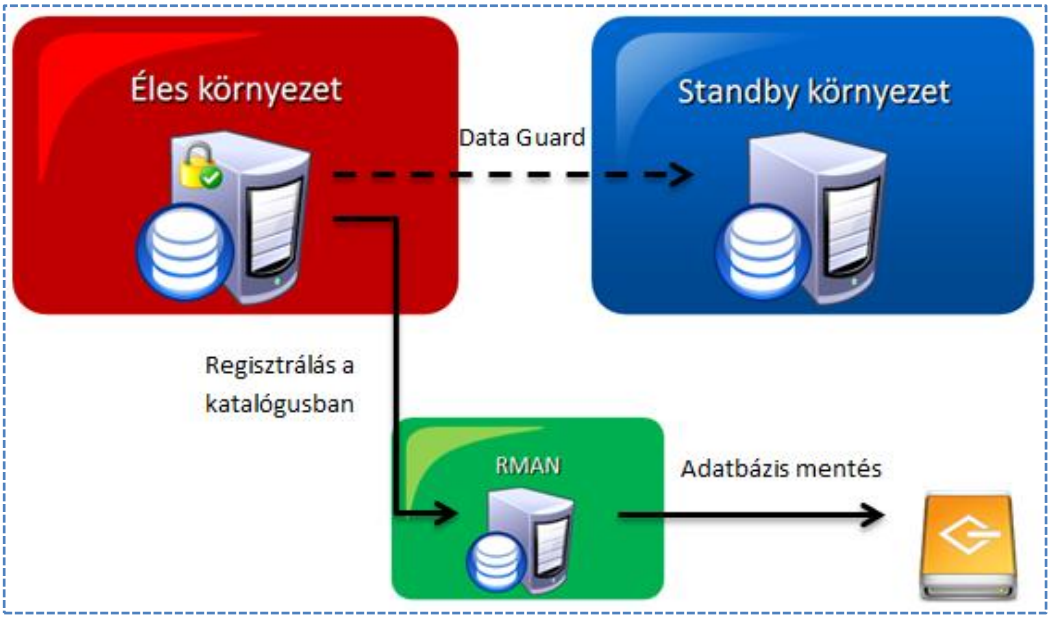
4. Az adatbázis-kezelő rendszer kialakítása

A dolgozatom szempontjából a legfontosabb rész következik most. Az elméleti felvezetésben röviden leírtakat próbálom meg a gyakorlatban megvalósítani egy konkrét működő rendszer keretében. Egy olyan rendszernek a keretében, amely megfelel egy középvállalat működése szempontjából fontos igényeknek, vagy akár egy egyetem tanulmányi rendszerének üzemeltetésének. Utóbbi esetben akár több száz, esetenként közel félezer felhasználóról beszélhetünk, akik egyidejűleg használják a rendszert. Ezeknek az elvárásoknak megfelelően igyekszem az adatbázis-kezelő környezetet kialakítani. Fontos leszögeznem, hogy eltekintek a kimondottan hardvereket, az operációs rendszert és az alkalmazásoldalt érintő kérdésektől, csak az adatbázis-kezelő rendszert érintő témakörökkel kívánok foglalkozni. Dolgozatomban az Oracle 11.2.0.1-es verziójú adatbázison mutatom be a legfontosabb beállításokat és megoldásokat, Windows alapokon.

A rendszer kialakításának fázisai a következőek:

- Az elsődleges (ORADB_P) adatbázis paramétereinek beállítása és az adatbázis elindítása.
- A mentésül és helyreállításul szolgáló katalógus tárolására szolgáló adatbázis (RMANDB) létrehozása és konfigurálása, valamint az elsődleges adatbázis (ORADB_P) regisztrálása a katalógusban.
- Az éles adatbázis (ORADB_P) mentésének végrehajtása.
- A másodlagos, azaz standby adatbázis (ORADB_SF) kialakítása a kapcsolat megteremtése az éles adatbázissal (ORADB_P). A kapcsolat ellenőrzése. (második standby adatbázis - ORADB_SL - kialakítása RMAN-nal)
- Az átkacsolási (switchover) teszt végrehajtása (ORADB_P és ORADB_SF között).
- Auditálási megoldások alkalmazása az éles adatbázison (ORADB_P).

Végezetül pedig, hogy vizuálisan is látható legyen az elérendő cél, az alább látható rendszer sémáját fogom kialakítani. Az éles adatbázis mellett két másik adatbázis is üzemelni fog az. Az egyik az adatbázis tükrözését fogja szolgálni, míg a másik a Recovery Manager helyreállítási katalógusát fogja tartalmazni.



7. ábra: A megvalósítandó rendszer modellje

4.1 A primary adatbázis konfigurálása

Minden adatbázis-adminisztrátor feladata, hogy az adatbázis-környezeteket az üzleti igényeknek megfelelően konfigurálja még a tényleges használat előtt. Ennek a konfigurálásnak sok szempontot kell figyelembe vennie: a technológiai és szoftveres környezetet, az üzleti elvárásokat, a hatékonyságot, a magas rendelkezésre állást és biztonságos üzemelést. Ezeknek a feltételeknek megfelelni sokszor nem könnyű, megvalósítása alapos tervezést igényel. Amennyiben sikerül egy optimális koncepciót kidolgozni, akkor az nagymértékben hozzájárulhat a hatékony, és eredményes működéshez. A következőkben olyan általános beállításokat ismertetek, amelyek segítségével megkönnyíthető az adatbázis menedzselése, valamint a performancia optimalizálása. A következő beállításokról lesz szó:

- Általános adatbázis paraméterek
- Performancia beállítások
- Audit alapbeállítások
- Undo és temp
- Táblaterek és adatfájlok

Az általános adatbázis paraméterek tárgyalása előtt kitérnék a 9-es verziótól elérhető szerver paraméter fájl (`spfile`) előnyeire. A korábbi paraméter fájlt (`pfile`) hivatott

leváltani ez az új megoldás. Ezekben az állományokban tárolódnak az adatbázis beállításai, ezekből olvassa fel az adatbázis a paramétereket. Az `spfile` újdonsága abban rejlik, hogy úgymond „menet közben” változtathatóak a benne lévő beállítások, amely változások a következő indításnál érvénybe lépnek, vagy adott esetben, ha `mount` állapotban van az adatbázis, akkor nemcsak a paraméterfájlban, hanem memória szinten is megváltoztathatóak egyes paraméterek. Ezzel szemben a paraméterfájl esetén mindig kézzel kell szerkeszteni az állományt és a változás csak memóriaszinten érvényesül. Biztonsági intézkedésként az `spfile` kézzel már nem szerkeszthető, így csak az adatbázison keresztül módosítható tartalma az `alter system` kezdetű parancsokkal.

Az adatbázis elindítása az első feladatunk az alapértelmezett paraméterfájl alapján:

```
SQL> startup mount
pfile='c:\oracle\product\11g\database\initoradb_p.ora'
```

Az adatbázis konfigurálása elején érdemes az alapbeállításokat tartalmazó paraméterfájlból az indításhoz szükséges szerver paraméterfájlt létrehozni a következő módon:

```
SQL> create spfile from pfile;
File created
```

Ezek után az adatbázis már alapértelmezettként az `spfile-t` fogja használni. Az `spfile` a `pfile-hoz` hasonlóan az $ORACLE_HOME/database könyvtárban található, és a következő névkonvenció alapján találhatjuk meg: `spfile<SID>.ora`, jelen esetben `spfileoradb_p.ora`.

Több, a szervertől függő paramétert is konfigurálhatunk, de ezek minden esetben mások lehetnek. Érdemes azonban foglalkozni a következő paraméterekkel: `db_block_size`, `db_unique_name`, `recycle_bin`, `remote_login_passwordfile`. A `recycle_bin` értelemszerűen egy lomtárhoz hasonló megoldás, a törölt objektumok, adatok ide kerülnek mielőtt a `purge` utasítással véglegesen törlődnének, vagy a `flashback` utasítással vissza nem kerülnek az eredeti helyükre. A környezet tervezésekor érdemes átgondolni, hogy szükséges-e ez a funkció, az üzleti igényeket ismerve. A `remote_login_passwordfile` paraméter igen fontos funkciót tölt be. Alapértelmezett értéke `EXCLUSIVE`. Ez biztosítja azt, hogy a `password file`-ban szereplő felhasználók akkor is tudjanak az adatbázishoz kapcsolódni, ha az éppen nem

fut, így megvalósítva egy bizonyos szintű autentikációt. Általános esetben a `password file` a SYS és SYSTEM felhasználókat tartalmazza. Új adatbázis készítésénél szintén szükséges a `password file` megléte. A következő utasítással hozható létre az ORAPWD funkció segítségével – 3 egyidejű bejelentkezéssel, valamint a korábbi fájl felülírásával:

```
> orapwd file = '$ORACLE_HOME\database\PWDoradb_p.ora' password =
<password> entries = 3 FORCE = Y;
```

Hogy kik jogosultak a `password file` alapján bejelentkezni az adatbázisba, azt a következő nézetből tudhatjuk meg: `SYS.V$PWFILE_USERS`. A `db_unique_name` az adatbázisok egyedi azonosíthatóságát biztosítja, a standby építésnél lesz jelentősége. A `db_block_size` a blokkméret beállítására hivatott. Ez a paraméter a telepítés után a későbbiekben nem változtatható meg. Bevált és elfogadott megoldás az, hogy az adattárházak 16 kilobyte-os, míg a tranzakció feldolgozó adatbázisok általában 8 kilobyte-os blokkméretet kapnak. Az adattárházak esetén azért indokolt ez az érték – többek között az optimalizáció érdekében – mert jelentős adatmennyiségen, hosszan tartó műveleteket végeznek általában, ami nagy mennyiségű blokkművelettel jár. Az optimalizáció megemlítése esetén nem szabad eltekinteni a memória kezelését érintő paraméterektől. Ezek nagyban függnek az adatbázis jellegétől, így beállításuk nagyfokú hozzáértést igényel. Egyszerűbb megoldás, ha az adatbázisra bízzuk a memória menedzselését – mind a *Server Global Area*-t (`SGA`), amely az adatbázishoz tartozó háttérfolyamatok memóriaterülete, mind a *Process Global Area*-t (`PGA`), amely az egyes felhasználók memóriaterületét jelenti. A következő táblázat mutatja meg, hogy milyen memóriakezelési módszerekkel konfigurálható az adatbázis.

Memória menedzselési mód	Az adatbázis adminisztrátor állítja be	Az adatbázis automatikusan kezeli
Automatikus memória menedzsment	▪ Az adatbázis instancia teljes memória mérete ▪ Instancia maximális memória mérete (opcionális)	▪ SGA méret ▪ SGA összetevőinek mérete ▪ PGA méret ▪ PGA összetevőinek mérete
Automatikus osztott memória-kezelés és automatikus PGA memória menedzsment	▪ SGA méret Instancia szintű PGA méret ▪ SGA maximális méret (opcionális)	▪ SGA összetevőinek mérete ▪ Egyéni PGA méretek

Egyéni osztott memória kezelés és automatikus PGA memória menedzsment	▪ SGA összetevőinek mérete ▪ PGA instancia szintű célmérete	▪ Egyéni PGA méretek

5. táblázat: Memóriakezelési módok (*Forrás: Oracle, 2010a*)

Én az adatbázis konfigurálásánál az automatikus memória menedzsmentet használom, azaz az adatbázisra bízom a rendelkezésre álló memória kezelését. Az általános beállítások végéhez közeledve érdemes még néhány szót szólni az auditálási alapbeállításokról. Az `audit_trail` paraméter DB-re állításával biztosítható, egy alapvető auditálási megoldás, amelynek során a `SYS.AUD$` táblában keletkeznek rekordok, amelyek a felhasználói aktivitásról tanúskodnak. Ennek hátránya, hogy sok felhasználó esetén rövid idő alatt, elég nagy helyet kezd elfoglalni ez az audit tábla, így kezeléséről gondoskodni kell. Megjegyzendő, hogy a DB érték az alapértelmezett beállítás. Az `audit_trail` nem érinti a SYSDBA, SYSOPER joggal rendelkező felhasználókat. Ha az ő tevékenységeiket is auditálni szeretnénk, hogy mit csinálnak, ha az adatbázis nincs elindítva, akkor az `audit_sys_operation` paramétert kell engedélyeznünk. Ekkor példaként említve a SYS felhasználó tevékenysége az operációs rendszer szinten kerül rögzítésre az audit fájlokban, ezáltal nyomon követhető a tevékenysége.

Végezetül néhány gondolat a táblaterekről és az azokhoz kapcsolódó paraméterekről. Alapvető koncepcionális elvárás, hogy az adatok tárolása során külön táblaterek kerüljenek kialakításra mind az adatok, mind az indexek számára, továbbá, ha lehetséges, fizikálisan is különböző diszkeken legyenek tárolva a hozzájuk tartozó adatfájlok. Így teljesítménybeli javulást érhetünk el. Célszerű – ha nem is Automatic Storage Management-et (ASM) – File System megoldást alkalmazni az állományok kezelésére operációs rendszer szinten. Így menedzselésük egyszerűbbé válik, mivel akár blokkmérettől függően gigabájtos nagyságrendű adatfájlok jöhetnek létre. (Linux vagy Unix operációs rendszeren a *raw device* megoldás – amikor diszk partíciókat hozunk létre, amelyeket az adatbáziskezelő-rendszer közvetlenül elérhet – ugyan teljesítményben jobb lehet, de nagyobb odafigyelést igényel, és összetettebb, bonyolultabb kezelhetőséggel jár együtt). Az adatokat tartalmazó táblatér létrehozásának módja a következő – SYS userrel, vagy DBA jogosultsággal rendelkező felhasználóval:

```
SQL> CREATE TABLESPACE APP_DAT_LRG DATAFILE
  'C:\ORACLE\ORADATA\ORADB_P\APPDATL01.DBF' SIZE 100M AUTOEXTEND ON
NEXT 10M MAXSIZE UNLIMITED
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO
FLASHBACK OFF;

Tablespace created
```

Hasonló módon hozható létre táblatér az indexeknek is. Célszerű ezt is megtenni. Táblaterek esetén érdemes még kitérni a `TEMP` és az `UNDO` táblatérre is. A temporary táblatér (`TEMP`) arra szolgál, hogy különböző műveleteket – amelyek elég sok memória területet igényelnének – ne a memóriában, hanem diszken végezzen el az adatbázis. Tipikusan ilyen műveletek közé tartoznak a rendezési műveletek, vagy pedig táblák összekapcsolásából eredő lekérdezések. Célszerű tehát vizsgálni, hogy az éles működés során az alkalmazások mekkora `TEMP` táblateret használnak és ennek figyelembevételével meghatározni a szükséges maximális méretét. Az `UNDO` táblatér hivatott a konzisztenciát biztosítani adatszinten. Az adatbáziskezelő-rendszer itt tárolja a változtatások – például `update`, `insert`, `delete` utasítás esetén – előtti állapotot, amely adott esetben visszaállítható a `rollback` utasítással. De funkcionalitása ennél többet takar. A táblatér betelése veszélyeztetheti az adatbázis működését is, így meg kell fontolni, mekkora mérettel hozzuk létre. Ugyanakkor közrejátszhat még, hogy mekkora időtartamra őrzi meg az adatbázis az adatokat az UNDO táblatérben. Ennek beállítására az `undo_retention` paraméter szolgál. A két órás időtartam beállítása a következő módon történik:

```
SQL> ALTER SYSTEM SET UNDO_RETENTION=7200 SCOPE=SPFILE;
System altered
```

Az RMAN és standby adatbázisok beállítása is fontos még az éles környezetben megkezdett működés előtt. A hatékony működés alapfeltétele egy jól átgondolt koncepció, ami a beállításokat illeti. A következőkben néhány RMAN-nal és standby adatbázissal kapcsolatos beállítási lehetőségekről lesz szó. Ezeket a paramétereket természetesen az éles rendszeren kell beállítani, amellett, hogy természetesen az RMAN-nak is megvannak a saját paraméterezési lehetőségei ugyanúgy, mint a Data Guard Command-Line Interface-nek (`DGMGRL`). Az RMAN-t érintő beállítások között a

db_recovery_file_dest, valamint a db_recovery_file_dest_size paraméterek lehetnek fontosak. Az első paraméter a flash_recovery_area helyét jelöli, míg a másik az adatbázis mentésének fizikai méretét határozza meg. Az alapértelmezett értékeket nem feltétlenül szükséges megváltoztatni. A flash_recovery_area-n belül viszont érdemes strukturálni az állományokat, ezt az RMAN beállításainak keretén belül tudjuk megtenni, amit majd a következő alfejezetben érintek. Egy másik érdekes beállítási lehetőség a 10g-s verziótól elérhető az adatbázis-adminisztrátorok számára. Ez a block change tracking, aminek a segítségével az adatbázis-mentések könnyíthetőek meg az RMAN számára. Engedélyezve ezt a beállítást, akkor az adatbázis egy logfájlban rögzíti az utolsó mentés óta történt változásokat blokkszinten. Így az RMAN-nak nem kell végignéznie mentés során az összes adatbázisblokkot, hanem a logfájl alapján sokkal gyorsabban elvégezheti a mentési folyamatot. A block change tracking-et beállító utasítás a következő:

```
SQL> ALTER DATABASE ENABLE BLOCK CHANGE TRACKING
USING FILE
'c:\oracle\flash_recovery_area\oradb_p\block_change_tracking\bct01.log
';

Database altered
```

Az standby adatbázist érintő paraméterek közül a legalapvetőbb az ARCHIVELOG mód beállítása. Enélkül nem is működhetne az adatbázis tükrözés folyamatát megvalósító Data Guard szolgáltatás. A következő utasításnak az alap adatbázis beállítási beállítások után kell közvetlenül következnie:

```
SQL> ALTER DATABASE ARCHIVELOG;

Database altered
```

A kialakított standby adatbázis-koncepció alapján érdemes az engedélyezés előtt először meghatározni, hova is kerüljenek az archived log fájlok. A következő utasításokat kell kiadnunk mount (startup mount) állapotban a beállítások rögzítése érdekében:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_FORMAT=' ORADB_%s_%r.%t.log'
SCOPE=SPFILE;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_1=
'LOCATION=c:\oracle\flash_recovery_area\oradb_p' SCOPE=SPFILE;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1=ENABLE SCOPE=SPFILE;
```

Az első utasítás értelemszerűen az archived log fájlok formai megkötéseit tartalmazza, ami a névkonvenciót illeti. Ezt követően egy második helyre történő logmásolás

beállítása és engedélyezése történik meg. Ez biztosítja a logfájlok továbbítását a standby oldal felé a Data Guard szolgáltatás segítségével. Az utolsó paraméter a minimális előírást adja meg, hogy hány helyre szükséges mindenképen a logfájlok másolása. Mount állapotban megnyitva az adatbázist, ezeket a parancsokat kiadva nem kell az adatbázist újraindítani. A rendszerünk jelenleg a konfigurálás után a következőképp ábrázolható a kiindulási állapotban:

8. ábra: Első fázis - Az éles adatbázis konfigurálása

Összefoglaló táblázat a beállított paraméterekről, nem érintve a memória-beállításokat:

Paraméter neve	Paraméter értéke	Leírás
db_block_size	8K	Az adatbázis blokkméretét meghatározó paraméter
db_unique_name	oradb_p	Az adatbázis egyedi nevét meghatározó paraméter
remote_login_passwordfile	exclusive	Egyedi password fájl, amely tartalmazza, hogy ki tud belépni az adatbázisba, minden esetben
recycle_bin	off	A „lomtár” kikapcsolását
audit_trail	db	Alapvető audit beállítás
audit_sys_operation	on	A SYS felhasználó tevékenységének monitorozása OS szinten
undo_retention	7200	Az undo táblatérben található adatok megőrzési időtartama másodpercben

db_recovery_file_dest	c:\oracle\flash_recovery_area	Az adatbázis mentéseinek helye
log_archive_format	ORADB_%s_%r.%t.log	Az archived logok névkonvencióját meghatározó beállítás
log_archive_dest_1	c:\oracle\flash_recovery_area\oradb_p	Az archived logok elsődleges helye

6. táblázat: Beállított adatbázis paraméterek

4.2. RMAN környezet kialakítása

4.2.1. Kezdeti lépések

Mint az már a korábbi fejezetből megtudhattuk, ahhoz, hogy az éles adatbázisunkról – vagy bármely adatbázisunkról – mentést tudjunk készíteni az Oracle Recovery Manager segítségével, szükségünk van egy külön adatbázisra, amely tartalmazza a helyreállítási katalógust (recovery catalog) és ez által mintegy repository-ként szolgál a folyamat során. Célszerű az RMAN adatbázist külön szerveren létrehozni, függetlenül a többi adatbázisunktól. Az adatbázis telepítése hasonlóan egy production, éles adatbáziséhoz történik: az egyszerűbb módszert alkalmazva a Database Configuration Assistant (DBCA) segítségével, vagy manuálisan a könyvtárszerkezet, az instancia létrehozásával és a szükséges scriptek lefuttatásával. Az adatbázis neve célszerűen legyen "RMANDB". Az Oracle adatbázis telepítése után érdemes néhány beállítást elvégezni. Egyrészt külön táblaterületet kell biztosítani az adatok tárolásához, másrészt egy technikai sémát is indokolt létrehozni, amely hivatott ellátni a szükséges műveleteket, valamint tartalmazza a szükséges objektumokat, adatokat. A táblatér létrehozása a következő módon történik a jelenleg még üres RMAN adatbázisban, SYS userrel:

```
SQLPLUS "sys@rmandb /as sysdba"

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production

SQL> CREATE TABLESPACE RMANTBS DATAFILE
  'C:\ORACLE\ORADATA\RMANDB\RMAN01.DBF' SIZE 50M AUTOEXTEND ON NEXT 1M
MAXSIZE 32767M
LOGGING
ONLINE
PERMANENT
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
BLOCKSIZE 8K
SEGMENT SPACE MANAGEMENT AUTO
```

```
FLASHBACK OFF;
Tablespace created
```

A táblatér az RMANTBS nevet kapta, meghatározott helyre került a hozzátartozó adatfájl, valamint kezdő és maximális mérete is meghatározásra került (autoextend-es táblatér mérete folyamatosan nőhet, amíg el nem éri a blokkméretből eredő maximális méretet). A következő utasítással a technikai séma jön létre az előzőleg létrehozott táblatér használatával. Alapvetően általános paraméterek alkalmazásával jön létre a séma, egyedül az előzőleg létrehozott táblatér alapértelmezettként beállítása tér el a default értékektől. Ezt követően pedig a jogosultsági beállítások elvégzése következik. Az alapvető jogok mellé a `RECOVERY_CATALOG_OWNER` role tartalmazza az összes jogosultságot, amik a helyreállítási katalógus hozzáféréséhez, menedzseléséhez szükségesek, ennél többet nem szükséges adni a sémának.

```
SQL> CREATE USER RMAN
  IDENTIFIED BY <rmanpassword>
  DEFAULT TABLESPACE RMANTBS
  TEMPORARY TABLESPACE TEMP
  PROFILE DEFAULT
  ACCOUNT UNLOCK;
  GRANT UNLIMITED TABLESPACE TO RMAN;
  ALTER USER RMAN QUOTA UNLIMITED ON RMANTBS;
User created

SQL> GRANT CONNECT, RESOURCE, RECOVERY_CATALOG_OWNER TO rman;
ALTER USER RMAN DEFAULT ROLE ALL;
Grant complete
```

Mindezek után a következő lépésként, mielőtt regisztrálnánk adatbázisunkat az RMAN adatbázisban, létre kell hozni a helyreállítási katalógust az RMAN adatbázisban, immáron az "`rman`" sémában:

```
>RMAN catalog=rman/<rmanpassword>@rmandb

Recovery Manager: Release 11.2.0.1.0 - Production
connected to recovery catalog database

RMAN> create catalog tablespace "RMANTBS";
recovery catalog created
```

Ezt követően nincs más dolgunk, mint regisztrálni az adatbázist a helyreállítási katalógusban, ennek módja, hogy egyszerre kapcsolódunk a katalógushoz és a

céladatbázishoz is, az alább látható módon – ezt a kapcsolódási módot a későbbiekben még többször használjuk. A kapcsolódás sikerességéről a DBID megjelenítése tanúskodik. A kapcsolódás után a "`register database`" paranccsal történik a metaadatok rögzítése a helyreállítási katalógusban. Ez a parancs csak akkor teljesül, ha egyedi DBID-vel rendelkezik a céladatbázisunk – ellenkező esetben hibát jelez számunkra az RMAN – valamint, ha eddig még nem volt regisztrálva az adatbázis a katalógusban. Az első hiba a DBID megváltoztatásával (DBNEWID utility alkalmazásával) orvosolható.

```
RMAN "catalog=rman/<rmanpassword>@rmandb
target=sys/<syspassword>@oradb_p"

Recovery Manager: Release 11.2.0.1.0 - Production

connected to target database: ORADB_P (DBID=479167586)
connected to recovery catalog database

RMAN> register database;

database registered in recovery catalog
starting full resync of recovery catalog
full resync complete
```

4.2.2. Általános beállítások

A következőkben áttekintésre kerülnek a legalapvetőbb Recovery Manager beállítások, amelyek segítségével testre szabhatjuk a mentési és helyreállítási folyamatokat. Röviden vagy elöljáróban a következőt kell leszögeznünk: az RMAN képes a target adatbázison utasításokat végrehajtani a csatlakozó felhasználó által. Ezek az utasítások lehetnek általános adatbázis-működést befolyásoló utasítások, mint például a shutdown vagy startup utasítások, de lehetnek SQL parancsok is. Továbbá akár utasítások egész sorát tartalmazó scriptek is futtathatók. A parancsok kiadásának módja összegezve a következő rövid példán látható. Ennek alkalmazása az elkövetkező fejezetekben látható lesz.

```
RMAN> run{
2> # utasitas
3> startup mount;
4> #
5> # sql utasítás
6> sql'alter database open';
7> #
8> # script
```

```
10>@$utvonal\rman_config.rman;
11>}
```

Eljutottunk odáig, hogy a céladatbázisunk, amelyről a mentést fogjuk készíteni, regisztrálásra és szinkronizálásra került a helyreállítási katalógusban. Mielőtt még elkezdenénk a teljes adatbázis mentését és esetleges helyreállítását, érdemes néhány RMAN specifikus beállítást áttekintenünk és szükség esetén beállítanunk. Az általános beállításokat az "rmandb" adatbázisból – catalog és target adatbázishoz egyaránt csatlakozva, mint a céladatbázis regisztrálásánál – a következő paranccsal tekinthetjük meg:

```
RMAN> show all;

RMAN configuration parameters for database with db_unique_name ORADB_P
are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK;
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F';
# default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; #
default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; #
default
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'c:\oracle\flash_recovery_area\Backup%d_DB_%u_%s_%p';
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT'
OPTIMIZE FOR LOAD TRUE ; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'C:\ORACLE\PRODUCT\11G\DATABASE\SNCFORADB.ORA'; # default
```

Amint az jól látható alapértelmezésben a legtöbb paraméter a default értéket veszi fel. Számunkra a következő paraméterek megváltoztatása, megemlítése érdemes:

- RETENTION POLICY TO REDUNDANCY
  - Ez a paraméter meghatározza, hogy az egyes adatfájlokból hány példány keletkezzen a mentés során. Amennyiben van rá lehetőség, érdemes ezt a paramétert 1-nél nagyobbra állítani.

- `DEFAULT DEVICE`
  - Ez a paraméter határozza meg, hogy hova történjen a mentés elvégzése. Ez alapesetben, mint látható, a disk-et jelenti, de lehetséges szalagos meghajtó megadása is. Esetünkben nem szükséges megváltoztatni.
- `CONTROLFILE AUTOBACKUP ON`
  - Ez a paraméter hivatott az adatfájlok mentése mellett a control állományok mentéséről is gondoskodni. A controlfile-ok mentései a `flash recovery area`-ban találhatóak meg. Mindenképp legyen bekapcsolva ez az opció.
- `DEVICE TYPE DISK PARALLELISM`
  - A parallelizmus mértékét beállító parancs. 1-nél nagyobb érték megadása esetén a parallelizmus fokának megfelelő számú csatornán történik a mentés folyamata.
- `CHANNEL DEVICE TYPE DISK FORMAT`
  - A mentés során keletkezett állományok elnevezésére ad opciókat. Itt megadhatunk konkrét mentési útvonalat és könyvtárat is. Érdemes számunkra konzekvensen kialakítani ezt a struktúrát.
- `ARCHIVELOG DELETION POLICY`
  - Ennek a beállításnak a segítségével meghatározható, hogy az `archived log` fájlok törlésre kerüljenek az összes `log_archive_dest` paraméter által meghatározott helyről, amennyiben bizonyos számú alkalommal megtörtént a lementésük a meghatározott helyre, szalagra, vagy lemezre. Ez a lehetőség hasznos lehet akkor, ha sok, nagyméretű `archived log` fájl keletkezik folyamatosan.

*(*Forrás: Oracle, 2010e*)*

A következő beállítások kerültek alkalmazásra a korábban felsorolt ismérvek alapján. Az RMAN mindegyik utasítás teljesítéséről visszajelzést ad. A script (`rman_config.rman`) fájl tehát a következőket tartalmazza:

```
################################
#Redundancia
CONFIGURE RETENTION POLICY TO REDUNDANCY 2;
#Optimalizacio
CONFIGURE BACKUP OPTIMIZATION ON;
#Controlfile elnevezese
```

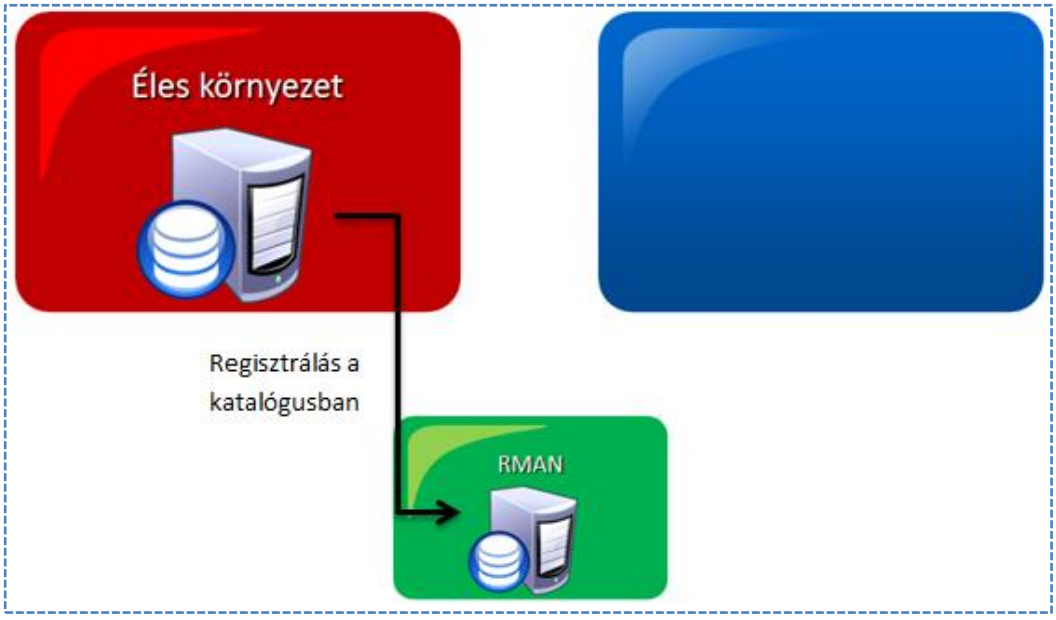
```
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO
'C:\ORACLE\flash_recovery_area\ctr_file\ctr_%F';
#Parallelizmus foka
CONFIGURE DEVICE TYPE DISK PARALLELISM 2 BACKUP TYPE TO BACKUPSET;
#Backup fajl elnevezese
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'c:\ORACLE\flash_recovery_area\Backup%d_DB_%u';
#Deletion policy
CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP 2 TIMES TO DISK;
################################
# File vege
################################
```

A script futtatása a `run{}` parancs által történik. Értelemszerűen egyaránt a katalógushoz és a céladatbázishoz is kapcsolódnunk kell egyúttal, mint azt már korábban láthattuk. A futás eredményéről a "`spool`" parancs által létrehozott szöveges fájlból tájékozódhatunk. Ez a parancs voltaképpen a képernyőre kerülő összes sort eltárolja egy fájlban, amíg a "`SPOOL LOG OFF`" paranccsal megállítjuk a folyamatot.

```
RMAN> SPOOL LOG to $oracle\scripts\rman_config.log'
RMAN> run{
2> @$oracle\scripts\rman_config.rman
3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15> 16> 17> 18> 19> 20> 21>
22> 23> 24> 25> 26> 27> 28> 29> 30> }
```

Az RMAN minden konfigurációs beállítás után szinkronizálja a helyreállítási katalógus metaadatait a céladatbázis kontrolfájljának adataival. Ez megtekinthető a fenti példa esetén is a logfile-ban.

Az eddig leírtak alapján a következőképpen alakul a rendszer sematikus ábrája:



9. ábra: Második fázis: RMAN adatbázis kialakítása

4.2.3. Offline és online backup

Az RMAN adatbázis környezetének beállítása, és magának az RMAN-nak a konfigurálása után elvégezhetjük immáron az éles adatbázis konzisztens és inkonzisztens mentését is. Ezeknek a mentési módoknak a tulajdonságait már korábban megismertük, most röviden összefoglalva: az offline – vagy más néven a cold – mentés lényege, hogy az adatokról konzisztens mentés készül. Ez csak abban az esetben lehetséges, ha az adatbázis-t nem használhatják a felhasználók. Erre szolgál az adatbázis megnyitása "`mount`" állapotban. Ekkor az adatbázis megnyitja és felolvassa a controlfile-t, amely tartalmazza az adatbázis fizikai adatstruktúráját. Így az RMAN képes lesz hozzáférni az adatfájlokhoz, táblaterekhez, logfájlokhoz. Ugyanakkor ebben az állapotban a felhasználók nem tudnak csatlakozni az adatbázishoz, kivétel persze a megfelelő jogosultsággal rendelkezők (SYS, SYSTEM). Az elméletbeli megvalósítás a gyakorlatban a következő script formájában történhet (`rman_cold_backup.rman`):

```
################################
# RMAN offline backup:
################################
# Target adatbazis leallitasa
shutdown immediate;
# Inditas mount modban
startup mount;
# Adatbazis backup
backup database;
# Controlfile backup
backup current controlfile;
# Adatbazis megnyitas
sql'alter database open';
################################
# File vege
################################
```

Az adatbázis leállítása és elindítása után a teljes adatbázis mellett a controlfile és az spfile is mentésre kerül. Az utasítás futtatása logolással a következőképpen néz ki, az előzőekben látottakhoz hasonlóan:

```
RMAN> SPOOL LOG to '$oracle\scripts\rman_cold_backup.log'
RMAN> run{
2> @$oracle\scripts\rman_cold_backup.rman
3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15> 16> 17> 18> 19> 20> 21>
22> 23> 24> 25> 26> 27> }
```

Az inkonzisztens mentés nagyon hasonló az offline mentéshez. Azonban ebben az esetben az adatbázis működése nem áll meg, a felhasználók ugyanúgy dolgozhatnak a rendszerben. Ebből is következik, hogy a valós állapotot tükröző mentés nem készülhet az adatbázisról, csak egy adott időpontra vonatkozó, ettől is lesz inkonzisztens. Elengedhetetlen feltétel, hogy az adatbázis "ARCHIVELOG" módban legyen és megfelelően legyen konfigurálva. Ebben az esetben a mentés a következő módon történik:

```
RMAN> SPOOL LOG to '$oracle\scripts\rman_hot_backup.log'
RMAN> run{
2> @$oracle\scripts\rman_hot_backup.rman
3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15> 16> 17> 18> }
```

Az script a következő parancsokat tartalmazza:

```
backup database plus archivelog;
#backup database plus archivelog delete input;
```

Ezek a parancsok hivatottak a teljes adatbázis, és mindemellett egyidejűleg az `archived log` fájlok mentését elvégezni. A második, végrehajtásra nem kerülő parancs esetén a logfájlok törlése is megtörténik egyúttal; hogy a kettő közül melyiket futtatjuk, az koncepcionális kérdés csupán. Amennyiben arra utaló hibával ér véget a script futtatása, hogy az RMAN nem találja a korábbi `archived log` fájlokat akkor a "`change … crosscheck`" utasítással ellenőrizhetjük az egyes fájlok státuszát. Jelen esetünkben ez így néz ki:

```
RMAN> change archivelog all crosscheck;
```

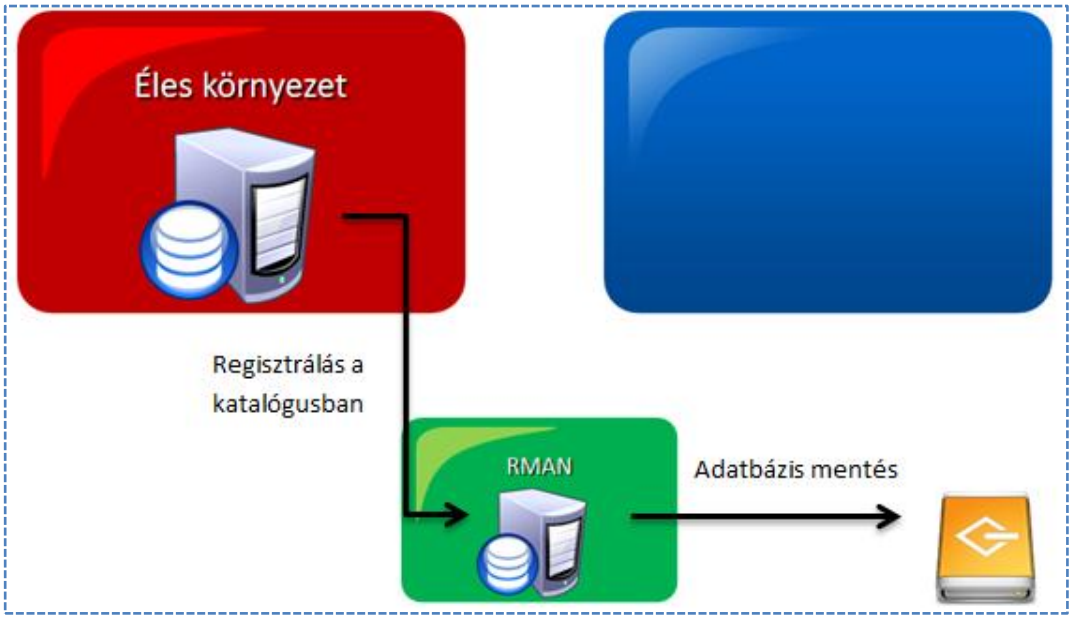
Az egyes fájlok hiánya esetén az RMAN töröltnek jelzi a fájlokat – az RMAN séma alatt található "RMAN.AL" táblában, a "STATUS" oszlopban ilyen esetben X érték szerepel – és a mentés során ezekkel nem foglalkozik. Ez a parancs ugyanígy alkalmazható mentések, másolatok fájljainak ellenőrzésére is. Érdemes naprakészen tartani az RMAN repository-t, a zökkenőmentes működtetés érdekében.

Fontos még említést tenni néhány egyéb lehetőségről, amelyet az RMAN biztosít számunkra, ha mentésekről esik szó. Mint láthattuk nem csak a teljes adatbázis mentése lehetséges, hanem emellett biztosított a controlfile és spfile mentése is, az archived log fájlok mentése is. De lehetőségünk van csupán táblaterek, vagy adatfájlok mentésére is, ami előnyös lehet, ha egy jól kialakított környezetben dolgozunk, így például, ha egy alkalmazás külön táblateret kap az adatok és indexek tárolására, az

előzőleg ismertetett koncepciók eredményeképp, akkor az érintett táblaterek mentése egyidejűleg az alkalmazás adatelemeinek és objektumainak a mentését is jelenti. Így elegendő ezeket a táblatereket visszatölteni alkalmazásoldali hiba esetén, megtakarítva ezzel időt és többletmunkát is. Erre a lehetőségre egy példa lehet a következő utasítás:

```
RMAN> backup tablespace system;
```

A rendszer a harmadik fázisban a következőképp ábrázolható:



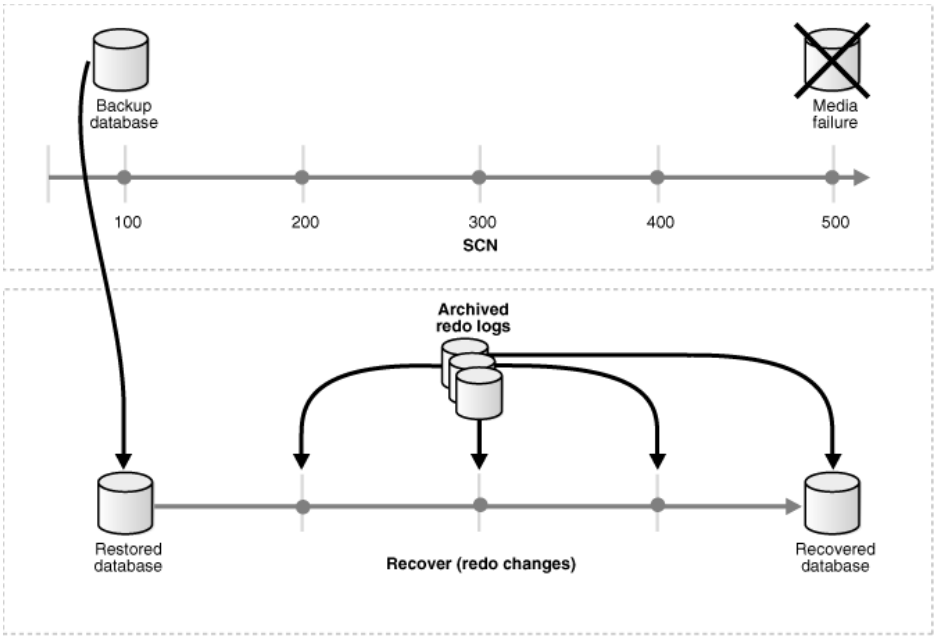
10. ábra: Harmadik fázis - Adatbázis mentés végrehajtása

4.2.4. Restore és Recover

Az adatbázis-mentések, legyenek azok offline vagy online mentések, abból a célból készülnek, hogy ha bekövetkezik a legnagyobb baj, akkor is gyorsan és problémamentesen helyre lehessen állítani az adatbázist. Az ideális az, ha az éles rendszereket sohasem kell visszatölteni, azonban ebben nem lehet bízni. Bármikor bekövetkezhet egy nem várt krízishelyzet. A nem éles rendszereknél viszont gyakori művelet lehet visszaállítani egy adott időpontra az adatbázis egészét vagy egyes részeit. A funkcionális és tesztkörnyezetek éppen ezt a célt szolgálják. Ezeken folynak a fejlesztések és tesztek, ahol elfogadott, hogy olyan hibák merülnek fel, ahol helyreállítás szükséges, hiszen ezeknek a környezeteknek ez a rendeltetésük. A Recovery Manager, mint látható volt egy egységes keretrendszert nyújt a mentések elkészítéséhez, de ami legalább olyan fontos, hogy a vissza-, és helyreállítások

folyamatait is menedzselhetővé teszi. Ennek segítségével és egy megfelelő koncepció kidolgozásával javítható a rendelkezésre állás, és az adatbiztonság kérdésköre is. Megemlíthető, hogy az adatbázis-mentések történhetnek az export/import szolgáltatás segítségével vagy a manuálisan adatfájlok lementésével, de ezek nem a „legszebb" megoldások, amikor rendelkezésünkre áll az RMAN szolgáltatás is.

Az előző fejezetben látható volt, hogyan menthető offline és online módon az adatbázis. Az éles adatbázis-kezelő rendszer (ORADB_P) `ARCHIVELOG` üzemmódban fut, annak érdekében, hogy a standby környezet megvalósítható legyen. Így erről az adatbázisról online mentés készülhet túlnyomó többségben, szemben egy tesztkörnyezetről, amelyről általában csak offline mentés szokott készülni. Az `ARCHIVELOG` üzemmód nem csupán a standby adatbázis-kezelő rendszer alapfeltétele. Az `archived log` fájlok segítségével az adatbázis nem csak visszaállítható (restore) az adatbázis mentésének specifikált időpontjára, hanem helyreállítható (recover) egy kívánt időpontig, hiszen ezek a fájlok tartalmazzák az összes változást, ami az adatbázisban történt. Ezt szemlélteti jól a következő ábra:



11. ábra: RMAN helyre-, és visszaállítás (*Forrás: Oracle, 2005, 15-7-es fejezet*)

Látható, hogy a meghibásodás időpontjáig is helyreállítható az adatbázis, szemben azzal a megoldással, amikor nem állnak rendelkezésre az `archive log` fájlok. Mivel az éles adatbázis (ORADB_P) ARCHIVELOG módban fut ezért az adatbázis bármilyen meghatározott időpontra helyreállítható.

Amennyiben vissza-, vagy helyreállításra van szükség egy adatbázis esetén, akkor az RMAN segítséget tud nyújtani a már meglévő mentések megjelenítésével. A bejelentkezés után a következő parancs segítségével megtudhatjuk milyen mentések vannak a céladatbázisunkról (ORADB_P) a helyreállítási katalógusban:

```
> RMAN catalog=rman/<rmanpwd>@RMANDB target=sys/<syspwd>@ORADB_P
connected to target database: ORADB_P (DBID=479167586)
connected to recovery catalog database

RMAN> list backup by file;
RMAN> list backup summary;
```

Ezekkel az utasításokkal átfogó képet kaphatunk arról, hogy milyen mentések készültek az adatbázisról és, hogy mikor készültek azok. Információ található a controlfile, az spfile, valamint az archived log fájlok mentéséről is. Az RMAN ezeken a parancsokon kívül még számos lekérdezésre ad lehetőséget. Érdemes megemlítést tenni arról a funkcióról is, hogy az RMAN információul szolgálhat az adatbázis adminisztrátornak, hogy mely fájlokról szükséges mentést készíteni a helyreállításhoz vagy, hogy mely állományok azok, amelyek nem visszaállíthatóak. Amennyiben a lekérdezések nem adnak vissza eredményt, akkor minden rendben van az adatbázissal az RMAN szerint. Ezek az utasítások a következőek:

```
RMAN> report need backup;
RMAN> report unrecoverable;
```

Az RMAN-nal adott a lehetőség, hogy visszaállítható legyen az egész adatbázis, egyes táblaterek, a controlfile, archived log fájlok, vagy az szerver paraméter fájl (spfile). Az RMAN automatikusan végrehajtja a visszaállítási műveletet, ha a megfelelő paranccsal erre utasítjuk. Ekkor a meglévő fájlokat felülírja, erre oda kell figyelni. (*Oracle, 2010d*)

A következő példában bemutatom a teljes adatbázis visszaállítás abban az esetben, ha az adatbázis `NOARCHIVELOG` módban van. Olyan esetet szimulálok, amikor lemez-meghibásodás következtében adatfájlok vesztek el vagy váltak használhatatlanná. Tipikusan tesztkörnyezetek, fejlesztői környezetek tartozhatnak ebbe a témakörbe. A visszatöltésekhez ezekben az esetekben akkor is szükség lehet, ha nem történt semmilyen meghibásodás, csupán az elvárt teszteredmények nem felelnek meg az előre meghatározott elvárásoknak. Első lépésként ellenőrizhető és

célszerű is ellenőrizni a következő utasítással, hogy a visszaállítás elvégezhető-e a korábbi mentések alapján:

```
RMAN> restore database validate;
```

Amennyiben hiba nélkül lefut a validálás, akkor hozzá lehet kezdeni a visszaállításhoz. Az céladatbázist el kell indítani nomount állapotban a hiba bekövetkezte után, majd ezt követően az RMAN-nal a következő parancsot kell végrehajtani:

```
RMAN> run{
2> restore database;
3> restore controlfile;
4> restore spfile;
5> }

Finished restore at 11-ÁPR.  -17
```

A controlfile helyreállítása ajánlott, az spfile-ra is szükség lehet adott esetben. Ennek a folyamatnak következtében az adatbázis teljesen abba az állapotba került, amikor a mentés készült. Az adatbázist ezután célszerű újraindítani mount állapotban, és megnyitni a következő utasításokkal:

```
SQL> recover database until cancel;

Specify log: {<RET>=suggested | filename | AUTO | CANCEL}
CANCEL

SQL> alter database open resetlogs;

Database altered
```

Ezt követően az adatbázis újra használhatóvá válik és egyúttal a visszaállítási folyamat véget ért.

Az `ARCHIVELOG` üzemmódban futó adatbázis helyreállítása nagyon hasonló módon történik, annyi különbséggel, hogy az archived log fájlokat is helyre kell állítani, amennyiben nincsenek meg. A visszaállítás menete megegyezik az előzőekben látottakkal a meghatározó utasítások röviden a következőek:

```
RMAN> restore database;

RMAN> restore archivelog all;

SQL> recover database;

SQL> alter database open resetlogs;
```

Ekkor az adatbázis a meghatározott időpontbeli mentés állapotát fogja felvenni, annak érdekében, hogy a nem várt esemény miatti visszaállítás a lehető legjobban tükrözze az akkori állapotokat az archived log fájlokat is vissza kell tölteni és rá kell görgetni az adatbázisra.

A fejezet végére érve rendelkezésünkre áll a konfigurált helyreállítási katalógus, a céladatbázisról elkészültek a mentések, amelyek akár szalagos meghajtóra is archiválhatók és visszatölthetők. Innentől kezdve gyakorlatilag bármikor visszaállítható, adott esetben pedig helyreállítható az adatbázis. A megfelelő beállítások és folyamatos mentések lehetőséget biztosítanak arra, hogy egy teljes mértékben megsemmisült adatbázist is vissza lehessen állítani az eredetihez nagyban hasonlító állapotba a rendelkezésre álló mentések alapján.

4.3. Standby adatbázis környezet kialakítása

A standby adatbázis környezet kialakítása az egyik legfontosabb szempont az üzletfolytonosság-, és rendelkezésre állás menedzsment témakörében, ami egy ilyen jellegű informatikai rendszer esetén szóba jöhet. Ennek standby adatbázis építésének lépéseit fogom a következőkben bemutatni, kétféle módon is. Előfeltételként rendelkezésre kell, hogy álljon az éles adatbázis-kezelő rendszer (ORADB_P), annak minden elemével és komponensével – mint például az adatbázismotorral, a konfigurált instanciával és listener-rel, amelyek másolás útján kerülnek majd a standby oldalra. Másrészt a standby oldalon szükséges a telepített Oracle adatbázismotor megléte. Mindezek után kezdhetünk hozzá az adatbázis környezet kialakításához.

4.3.1. Konfiguráció

A standby adatbázis (ORADB_SF) létrehozásához néhány változtatást kell megvalósítani az eddig konfigurált éles adatbázison. Ezeknek célja, hogy a két adatbázist szinkronban lehessen tartani és meg lehessen valósítani szükség esetén az átkapcsolást a két oldal között. Az `ARCHIVELOG` mód már korábban bekapcsolásra került, így ezzel most nem kell foglalkozni. Viszont ahhoz, hogy minden változás rögzüljön a redo log fájlokban és az azokból létrejövő `archived log` fájlokban, szükséges a következő utasítással erre kényszeríteni az adatbázist, `mount` állapotban:

```
SQL> ALTER DATABASE FORCE LOGGING;
Database altered
```

A következő lépés ezután a standby logfile csoport létrehozása. Arról, hogy jelenleg milyen logfájlok vannak a `SELECT * FROM V$LOGFILE;` lekérdezés adhat információt.

Mivel 1-es, 2-es és 3-as csoport már létezik, ezért 4-től 6-ig lesznek a standby logfájlok. A létrehozásuk a következőképp történik. (*Oracle, 2010c*)

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE GROUP 4 SIZE 50M;
SQL> ALTER DATABASE ADD STANDBY LOGFILE GROUP 5 SIZE 50M;
SQL> ALTER DATABASE ADD STANDBY LOGFILE GROUP 6 SIZE 50M;

Database altered
```

A következő lépés a standby környezethez szükséges paraméterek rögzítése az adatbázis paraméterfájljában. Ez történhet `spfile`, de `pfile` szinten is, de mindkét esetben ez a művelet az adatbázis újraindítását igényi. Célszerű a `pfile`-t használni, mert ennek kézi szerkesztése jóval gyorsabb, mint az `spfile`-ban beállítani az adatbázison keresztül. Új, a jelenlegi beállításokat tükröző `pfile` készítése a `'CREATE PFILE FROM SPFILE;'` utasítással lehetséges. A létrejött fájlban az alábbi sorokat kell hozzáadni, illetve szerkeszteni:

```
*.control_files='c:\oracle\oradata\oradb_p\control01.ctl','c:\oracle\f
lash_recovery_area\oradb_p\controlfile\control02.ctl'
*.instance_name='ORADB_P'
*.db_name='ORADB_P'
*.service_names='ORADB_P'
*.db_unique_name='ORADB_P'
*.log_archive_config='dg_config=(ORADB_P,ORADB_SF)'
*.FAL_CLIENT='ORADB_P'
*.FAL_SERVER='ORADB_SF'
*.log_archive_dest_1='LOCATION=c:\oracle\flash_recovery_area\ORADB_P
VALID_FOR=(ALL_LOGFILES,ALL_ROLES) DB_UNIQUE_NAME=ORADB_P'
*.log_archive_dest_2='SERVICE=ORADB_SF ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=ORADB_SF'
*.log_archive_dest_state_1='enable'
*.log_archive_dest_state_2='enable'
*.log_archive_max_processes=15
*.DB_FILE_NAME_CONVERT='c:\oracle\oradata\ORADB_P','c:\oracle\oradata\
ORADB_SF'
*.log_file_name_convert='c:\oracle\oradata\ORADB_P\','c:\oracle\oradat
a\ORADB_SF\'
```

Az adatbázis azonosítását szolgálja az első négy paraméter, a 11-es verziótól kötelező a `db_unique_name` paraméter használata is. A többi paraméter a kapcsolat megteremtését szolgálja a két adatbázis között. A konfigurációért, a logfájlok átviteléért a Data Guard szolgáltatás a felelős (`log_archive_config`), a szerepek beállítása után (`FAL_`) az elsődleges és másodlagos logfájl elérési útvonalainak megadása következik. Az elsődleges helyet az adatbázis saját számára használja, míg a második hely, mint az látható is, egy szolgáltatást takar, amely hivatott a standby adatbázis számára továbbítani a megfelelő logfájlokat. A Data Guard ebből a

paraméterből ismeri fel, hogy hova kell másolnia az állományokat. A convert-tel kezdődő beállítások a két adatbázis közötti kompatibilitásról gondoskodik.

Ezt követően az adatbázis a megszerkesztett paraméterfájl alapján kell elindítani az elsődleges adatbázist. Ennek módja a következő:

```
SQL> startup mount
pfile='c:\oracle\product\11g\database\initoradb_p.ora'
```

A standby adatbázis számára szükséges controlfile-t is készíteni, amely tartalmazza a szükséges információkat az adatbázisról, így a fizikai adatállományok helyét is. Erre külön specifikált utasítás létezik, amely a következő:

```
SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS
'C:\ORACLE\ORADATA\ORADB_SF\CONTROL01.CTL';
```

Ezt a fájlt át kell másolni két példányban a standby oldalra, a megfelelő elérési útvonalra annak érdekében, hogy a standby adatbázis meg tudjon nyílni mount állapotban. Ekkor létrehozhatjuk egyúttal a megfelelő könyvtárstruktúrát is. A flash_recovery_area-n belül létre kell hozni egy ORADB_SF, majd abban egy CONTROLFILE könyvtárat, valamint Az ORADATA alatt szintén létre kell hozni egy ORADB_SF mappát. A standby controlfile mellett át kell másolni az éles adatbázis password fájlját is annak érdekében, hogy be tudjunk jelentkezni az adatbázisba. Az új adatbázis eléréséhez meg kell szerkesztenünk a kapcsolódáshoz elengedhetetlen listener konfigurációs fájljait. A `listener.ora` fájlban definiálható új listener, a `tnsnames.ora` fájlban pedig a kapcsolati azonosítót kell beállítani. Ügyelni kell a port kiosztásra és a szerviz pontos megnevezésére, különben nem lehetséges majd csatlakozni az adatbázishoz.

`listener.ora` fájlban:

```
SID_LIST_LISTENER_ORADB_SF =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = ORADB_SF)
      (ORACLE_HOME = c:\oracle\product\11g)
      (GLOBAL_DBNAME = ORADB_SF)
    )
  )

LISTENER_ORADB_SF =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1552))
      (ADDRESS = (PROTOCOL = TCP)(HOST = server2)(PORT = 1552))
    )
  )
```

tnsnames.ora fájlban:

```
ORADB_SF =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = server2)(PORT = 1552))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ORADB_SF)
    )
  )
```

A fenti paraméterek alapján válik elérhetővé mindenki számára az adatbázis. Listener-t egyúttal el is indíthatjuk, ha belépünk az LSNRCTL alkalmazásba, vagy parancssori utasításon keresztül meghívjuk a szükséges utasítást. Ez egy ugyan olyan Oracle szolgáltatás, mint az oradim, vagy az orapwd, ennek kimondottan szerepe a listenerek konfigurálásában van.

```
LSNRCTL> start listener_oradb_sf;
The command completed successfully
```

Az előkészületek végéhez közeledve Windows környezeten létre kell hozni manuálisan az adatbázis instanciát a standby oldalon.

```
> oradim –new –sid ORADB_SF –startmode AUTO
```

Ezzel az utasítással létrejön az új instancia, az `oradim` szolgáltatás segítségével. Ezzel egyidejűleg az előkészületek befejeződtek. A következő lépés az adatbázis fizikai másolatának elkészítése a standby oldalon, majd az adatbázis tükrözés folyamatának elindítása.

Az standby adatbázis (ORADB_SF) fizikai kialakításának kétféle módja is lehetséges. Az egyik a Recovery Manager használata, a másik pedig a manuális úton történő létrehozás. Az RMAN alkalmazása megkönnyíti a folyamat véghezvitelét, azonban nagyfokú figyelmet igényel a helyes paraméterek beállítása. Ebben az esetben gyakorlatilag ugyan az a folyamat történik, mint ha manuálisan történne az adatbázis kialakítása. Ezért először bemutatom, hogyan történik kézzel a standby adatbázis létrehozása, majd ezután kitérek az RMAN esetére is.

Természetesen az instancia elindításához a standby oldalon is egy testre szabott paraméterfájl szükséges. Érdemes az éles adatbázis paraméterfájlját megszerkeszteni és átmásolni a másodlagos oldalra. Az alábbi sorokat kell módosítani az itt látható módon:

```
*.control_files='c:\oracle\oradata\oradb_sf\control01.ctl','c:\oracle\
flash_recovery_area\oradb_sf\controlfile\control02.ctl'
*.instance_name='ORADB_SF'
*.db_name='ORADB_P'
*.service_names='ORADB_SF'
*.db_unique_name='ORADB_SF'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=ORADB_SFXDB)'
*.log_archive_config='dg_config=(ORADB_P,ORADB_SF)'
*.FAL_CLIENT='ORADB_SF'
*.FAL_SERVER='ORADB_P'
*.log_archive_dest_1='LOCATION=c:\oracle\flash_recovery_area\ORADB_SF
VALID_FOR=(ALL_LOGFILES,ALL_ROLES) DB_UNIQUE_NAME=ORADB_SF'
*.log_archive_dest_2='SERVICE=ORADB_P ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=ORADB_P'
*.DB_FILE_NAME_CONVERT='c:\oracle\oradata\ORADB_P','c:\oracle\oradata\
ORADB_SF'
*.log_file_name_convert='c:\oracle\oradata\oradb_P\','c:\oracle\oradat
a\oradb_SF\'
```

Amint az látható a `db_name` paraméteren kívül a többi azonosító megváltozott, a logfájlok helyét meghatározó paraméterek is felcserélődtek értelemszerűen. Ez a konfiguráció fogja biztosítani, hogy átkapcsolás esetén is működőképes legyen megfelelő módon mindkét adatbázis.

Az adatfájlok átmásolásával érhető el, hogy fizikálisan megegyezzen a két adatbázis egymással. Az átmásolás történhet úgy, hogy az éles adatbázis üzemel, de történhet úgy is, hogy leállítjuk a feladat végrehajtásának idejére. Amennyiben nem állítjuk le, akkor gondoskodni kell arról, hogy az adatfájlok konzisztensek maradjanak, ne változzon tartalmuk a folyamat alatt. Ennek érdekében a következő utasításokat kell végrehajtani:

```
SQL> alter system switch logfile;
System altered
SQL> alter database begin backup;
Database altered
```

Az első utasítással kényszeríthető az adatbázis arra, hogy a redo logfájlokban szereplő adatokat azonnal írja ki archived log fájlokba. Célszerű ezt végrehajtani. A második utasítás hivatott az előbb említett konzisztenciát megőrizni. Az utasítás végrehajtásával az összes változást eltárolja az adatbázis és nem módosítja az adatfájlokat addig, amíg `begin backup` módban van az adatbázis. Ebben az állapotban lemásolható az éles adatbázis úgy, hogy közben nem kell leállítani azt.

Az adatfájlok másolása után a következőt kell tenni, hogy befejezzük a backup üzemmódot:

```
SQL> alter database end backup;
Database altered
```

Korábban, mint arról már volt szó, létrejött a megfelelő könyvtárstruktúra, a password file és a controlfile is a helyén van, beállításra került a standby adatbázis paraméterfájlja. A listener is a megfelelő módon van konfigurálva és el is lett indítva. Nincs más hátra, mint elindítani a standby adatbázist a paraméterfájlja alapján, majd spfile-t létrehozni. Újraindítani az adatbázist, hogy az spfile-t használja alapértelmezettként, ezután pedig elindítani a logfájlok átvitelének folyamatát. Az megfelelő utasítások sorrendben a következőek:

```
>sqlplus "sys/<syspwd>@oradb_sf /as sysdba"
Connected to an idle instance.

SQL> startup nomount
pfile='c:\oracle\product\11g\database\INIToradb_sf.ora'
ORACLE instance started.

SQL> create spfile from
pfile='c:\oracle\product\11g\database\INIToradb_sf.ora';
File created

SQL> shutdown immediate
ORACLE instance shut down

SQL> startup open read only
ORACLE instance started
Database mounted
Database opened
```

Amint az látható, az adatbázis sikeresen megnyílt, csak olvasható üzemmódban. Ez szükséges annak érdekében, hogy ne történjenek benne változások addig, amíg el nem indul a tükrözés folyamata. A tükrözés elindítása előtt érdemes ellenőrizni az adatbázisok jelenlegi állapotát a megfelelő lekérdezések segítségével.

Az éles adatbázisban (ORADB_P):

```
SQL> select name, open_mode, protection_level, database_role from
v$database;
NAME      OPEN_MODE            PROTECTION_LEVEL     DATABASE_ROLE
--------- -------------------- -------------------- ----------------
ORADB_P   READ WRITE           MAXIMUM PERFORMANCE  PRIMARY
```

```
SQL> select name, sequence# from v$archived_log;

NAME                                                                SEQUENCE#
------------------------------------------------------------------ ---------
C:\ORACLE\FLASH_RECOVERY_AREA\ORADB_P\ORADB_14_748229602.1.LOG              14
C:\ORACLE\FLASH_RECOVERY_AREA\ORADB_P\ORADB_15_748229602.1.LOG              15
C:\ORACLE\FLASH_RECOVERY_AREA\ORADB_P\ORADB_16_748229602.1.LOG              16
C:\ORACLE\FLASH_RECOVERY_AREA\ORADB_P\ORADB_17_748229602.1.LOG              17
C:\ORACLE\FLASH_RECOVERY_AREA\ORADB_P\ORADB_18_748229602.1.LOG              18
C:\ORACLE\FLASH_RECOVERY_AREA\ORADB_P\ORADB_19_748229602.1.LOG              19
C:\ORACLE\FLASH_RECOVERY_AREA\ORADB_P\ORADB_20_748229602.1.LOG              20
C:\ORACLE\FLASH_RECOVERY_AREA\ORADB_P\ORADB_21_748229602.1.LOG              21
```

A standby adatbázisban (ORADB_SF):

```
SQL> select name, open_mode, protection_level, database_role from
v$database;

NAME      OPEN_MODE            PROTECTION_LEVEL     DATABASE_ROLE
--------- -------------------- -------------------- ----------------
ORADB_P   READ ONLY            MAXIMUM PERFORMANCE  PHYSICAL STANDBY

SQL> select sequence#, applied from v$archived_log;

SEQUENCE# APPLIED
---------- ---------
        21 YES
```

Amint az látható, a lekérdezések alapján minden rendben van a két adatbázissal, az adatbázis neve mindkét esetben ugyanaz, valamint a role-ok is megfelelőek. A logfájlok szekvenciája is megegyezik. Ezek alapján elindítható a tükrözés folyamata:

```
SQL> alter database recover managed standby database disconnect from
session;
Database altered
```

Az utasítás következtében elkezdődik a logfájlok továbbítása a standby oldal felé, ahol ezek az archived logok feldolgozásra kerülnek, így bármi változás történik az éles oldalon, az pillanatokon belül megjelenik a standby oldalon is. Ennek ellenőrzése hasonlóan az előzőekhez:

Éles oldalon (ORADB_P):

```
SQL> alter system switch logfile;
System altered

SQL> select sequence# from v$archived_log where sequence# > 20;

SEQUENCE#
----------
        21
        22
```

Standby oldalon (ORADB_SF):

```
SQL> select sequence#, applied from v$archived_log

 SEQUENCE# APPLIED
---------- ---------
        21 YES
        22 IN-MEMORY
```

Majd pár másodperc múlva:

```
SQL> select sequence#, applied from v$archived_log;

 SEQUENCE# APPLIED
---------- ---------
        21 YES
        22 YES
```

Ezzel gyakorlatilag a folyamat lezárult, megvalósult az adatbázis tükrözése. A logok továbbítódnak a standby oldal felé, ahol azokat az adatbázis fel is dolgozza. Az átkapcsolás a két oldal között megvalósítható.

Második standby adatbázis létrehozása RMAN-nal

A standby adatbázis építése, mint azt említettem, RMAN-nal is megvalósítható. Az előkészületek után a következő a teendő: csatlakozni kell mind az éles, mind a standby adatbázishoz RMAN-nal és az adatbázis másolását biztosító utasítást kell kiadni. Az RMAN-nal történő standby adatbázis kialakítását az ORADB_SL elnevezésű adatbázison mutatom be. A folyamat a standby adatbázis elindításától kezdve – ebben az esetben egy teljesen alapbeállításokat (memória, elérési útvonalak) tartalmazó pfile alapján – tér el a manuális megvalósítástól. Előfeltétel a konfigurált éles adatbázis a korábban látott pfile alapján, a könyvtárszerkezet kialakítása, a password file megléte, a listener konfigurálása. Standby controlfile-t ebben az esetben nem kell készíteni, mivel ezt az RMAN elvégzi helyettünk. Azonban az adatbázis létrehozása előtt néhány paramétert meg kell változtatni az éles (ORADB_P) adatbázison, ha két standby adatbázis üzemel egyszerre. A következőt kell tenni ennek érdekében:

```
alter system set
log_archive_config='dg_config=(ORADB_P,ORADB_SF,ORADB_SL)';

alter system set log_archive_dest_3='SERVICE=ORADB_SL ASYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=ORADB_SL';

alter system set log_archive_dest_state_3=enable;

alter system set
db_file_name_convert='c:\oracle\oradata\ORADB_P','c:\oracle\oradata\OR
ADB_SF','c:\oracle\oradata\ORADB_P','c:\oracle\oradata\ORADB_SL'
scope='spfile';
```

```
alter system set
log_file_name_convert='c:\oracle\oradata\ORADB_P','c:\oracle\oradata\O
RADB_SF','c:\oracle\oradata\ORADB_P','c:\oracle\oradata\ORADB_SL'
scope='spfile';

System altered
```

Eljutva a megvalósítási fázishoz az alábbi utasításokat kell végrehajtani, ebben a sorrendben:

```
RMAN> connect target sys/<syspwd>@ORADB_P

connected to target database: ORADB_P (DBID=479167586)

RMAN> connect auxiliary sys/<syspwd>@ORADB_SL

connected to auxiliary database: ORADB_SL (not mounted)

RMAN> SPOOL LOG to '$oracle\scripts\create_oradb_sl_with_rman.log'

RMAN> @c:\oracle\create_oradb_sl_with_rman.sql
RMAN> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15> 16> 17> 18> 19>
20> 21> RMAN> RMAN>
```

A script a következő utasítást tartalmazza:

```
run {
    allocate channel p1 type disk;
    allocate channel p2 type disk;
    allocate channel p3 type disk;
    allocate channel p4 type disk;
    allocate auxiliary channel s1 type disk;
    duplicate target database for standby from active database
    spfile
        parameter_value_convert 'ORADB_P','ORADB_SL'
        set db_unique_name='ORADB_SL'
        set
db_file_name_convert='c:\oracle\oradata\ORADB_P','c:\oracle\oradata\OR
ADB_SF','c:\oracle\oradata\ORADB_P','c:\oracle\oradata\ORADB_SL'
        set
log_file_name_convert='c:\oracle\oradata\ORADB_P','c:\oracle\oradata\O
RADB_SF','c:\oracle\oradata\ORADB_P','c:\oracle\oradata\ORADB_SL'
        set
control_files='c:\oracle\oradata\ORADB_SL\control01.ctl','c:\oracle\fl
ash_recovery_area\ORADB_SL\CONTROLFILE\control02.ctl'
        set log_archive_max_processes='15'
        set fal_client='ORADB_SL'
        set fal_server='ORADB_P'
        set standby_file_management='AUTO'
        set log_archive_config='dg_config=(ORADB_P,ORADB_SF,
ORADB_SL)'
        set log_archive_dest_2='service=ORADB_P ASYNC
valid_for=(ONLINE_LOGFILE,PRIMARY_ROLE) db_unique_name=ORADB_P'
     ;
     }
```

Gyakorlatilag nagyon hasonló módon hozza létre az RMAN a standby adatbázist, mint az előzőekben bemutatott manuális módszer. Különböző csatornákat kell megadni, amelyeken keresztül dolgozhat, valamint az spfile paraméter beállításait is meg kell határozni. Az utasítás kulcsa a `duplicate target database for standby from active database` sor. A logfáljban láthatjuk a folyamat teljes lefutását. Az eredményt célszerű ellenőrizni ebben az esetben is, az eddig megszokott módon, az ORADB_SL adatbázisban:

```
SQL> select name, open_mode, protection_level, database_role from
v$database;
NAME      OPEN_MODE            PROTECTION_LEVEL     DATABASE_ROLE
--------- -------------------- -------------------- ----------------
ORADB_P   MOUNTED              MAXIMUM PERFORMANCE  PHYSICAL STANDBY
```

Ugyan ez a megoldás gyorsabb, mint a manuális módszer, azonban sokkal nagyobb odafigyelést igényel. Két standby adatbázis esetén nem szabad elfeledkezni a `log_archive_dest` paraméterek újrakonfigurálásáról minden egyes adatbázis esetén, abból a célból, hogy ha átkapcsolás történik valamelyik másodlagos adatbázisra, akkor a logok átvitele továbbra is zavartalanul működjön minden standby oldal felé.

Egy újraindítást és ellenőrzést követően kiadható a tükrözést elindító utasítást.

```
SQL> alter database recover managed standby database disconnect from
session;
System altered
```

Alternatív megoldásként amennyiben nem szeretnénk, ha bizonyos okokból egyszerre üzemelne két azonnal frissülő standby adatbázis, akkor alkalmazhatunk egyéb lehetőségeket is. Egy sok változási állományt generáló éles adatbázisnak, ha két helyre kell továbbítania a logfájlokat, akkor az jelentős hálózati terheléssel, és lemezigénnyel járhat. A lemezigény kezelhető, ha a logfájlokat bizonyos idő után szalagos meghajtóra archiválhatók és törölhetők a diszkekről, azonban a hálózati forgalom növekedése még mindig fennáll. Ha az adatbázis környezet nem igényli két gyorsan frissülő standby adatbázis egyidejű működését, akkor az egyik standby lehet kevésszer, specifikált időpontban frissülő adatbázis is, amelyet használhatnak akár speciális célokra is. Dolgozatomban ennek a lehetőségnek a beállítását is bemutatom. Lassan frissülő adatbázisnak megfelel az előzőekben specifikált RMAN alkalmazásával létrejött standby (ORADB_SL). A görgetéshez célszerű egy olyan időpontot választani, amikor a

standby logok átvitele és feldolgozása nem veszélyeztet más folyamatokat, performancia szempontokból. Ez lehet akár egy éjszakai időpont is, vagy napközben dél körül, amikor kevesen használják a rendszert. Ekkor a következő utasítássorozatot lehetséges kiadni, lehetőleg automatizáltan, script formájában:

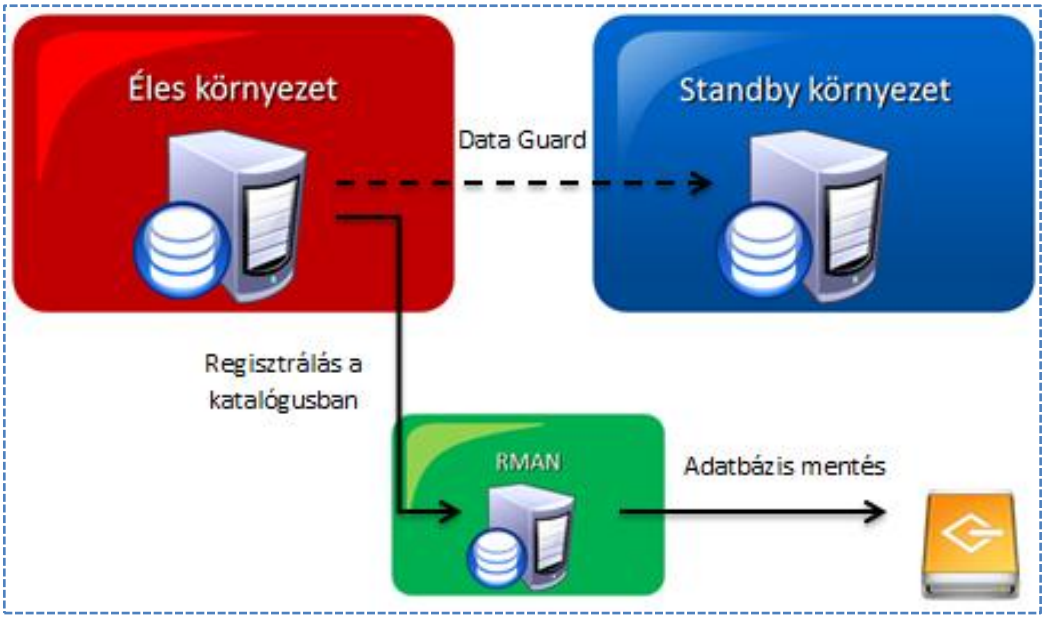
```
> sqlplus "sys/<syspwd>@oradb_sl / as sysdba"

SQL> shutdown immediate
SQL> startup nomount
SQL> alter database mount standby database;
SQL> alter session set nls_date_format='yyyy.mm.dd hh24:mi:ss';
SQL> SET AUTORECOVERY ON
SQL> RECOVER AUTOMATIC STANDBY DATABASE until time '$UNTILTIME'
parallel 4;

SQL> shutdown immediate
SQL> startup nomount
SQL> alter database mount standby database;
SQL> alter database open read only;
```

Az utasítássorozat a standby adatbázis (ORADB_SL) újraindítása után beállítja a dátumformátumot, majd automatikusra állítja a logfájlok feldolgozását. A 6. sorban szereplő parancs segítségével lehetséges a standby adatbázist az `$UNTILTIME` változóban meghatározott időpontig görgetni. Amint ez a folyamat végzett egy újraindítás után megnyitható az adatbázis csak olvasható üzemmódban, így lekérdezésekhez, riportoláshoz, speciális feladatok elvégzésére alkalmas lesz. Mindez automatizálva rendkívül hasznos lehet üzletfolytonosság menedzsment, performancia menedzsment és az üzleti elvárások szempontjából.

Összefoglalásul mindezek után megállapíthatjuk, hogy a rendszer egy újabb jelentős fázist lépett előre. Kiemelten üzletfolytonosság-, és rendelkezésre állás menedzsment szempontjából. A bemutatottak alapján tehát már egy standby adatbázis (vagy akár kettő) is rendelkezésre áll. A rendszer sémája a következőképp alakul ettől a fázistól kezdve:



12. ábra: Negyedik fázis - Standby környezet kialakítása

4.3.2. Switchover teszt

A tervezett és nem tervezett leállások következtében azzal kell számolni, hogy a rendszer nem elérhető rövidebb vagy hosszabb ideig. Ekkor értelemszerűen senki nem tudja használni a rendszert, nem lehet elvégezni a napi üzleti folyamatokat. Ha ez éppen rossz időben történik, akkor komoly üzleti hatással járhat. Ennek a kockázatnak kiküszöbölésére és menedzselésére, a rendelkezésre állás növelése érdekében létrejött a standby adatbázis, amit az előzőekben mutattam be. A teljes körű bevezetés és használat előtt célszerű végrehajtani az átkapcsolási tesztet az éles (ORADB_P) és a standby (ORADB_SF) adatbázis között, hogy látható legyen minden rendben működik-e. Amennyiben a teszt sikeres, akkor a tényleges üzembe helyezés is lehetséges. A teszt során figyelemmel kell kísérni az egyes adatbázisok beállításait, hogy ezek megfeleljenek az elvárásoknak. A kezdeti paraméterek a következők:

Éles rendszer (ORADB_P):

```
SQL> select name, open_mode, protection_level, database_role from
v$database;

NAME      OPEN_MODE            PROTECTION_LEVEL     DATABASE_ROLE
--------- -------------------- -------------------- ----------------
ORADB_P   READ WRITE           MAXIMUM PERFORMANCE  PRIMARY
```

```
SQL> select db_unique_name, database_role, switchover_status from
v$database;

DB_UNIQUE_NAME                 DATABASE_ROLE    SWITCHOVER_STATUS
------------------------------ ---------------- --------------------
ORADB_SF                       PRIMARY          TO STANDBY
```

Standby rendszer (ORADB_SF):

```
SQL> select name, open_mode, protection_level, database_role from
v$database;

NAME      OPEN_MODE            PROTECTION_LEVEL     DATABASE_ROLE
--------- -------------------- -------------------- ----------------
ORADB_P   READ ONLY            MAXIMUM PERFORMANCE  PHYSICAL STANDBY
```

Amint az látható az egyes szerepek megfelelőek, ha meggyőződtünk arról, hogy senki nem használja aktívan az adatbázist, akkor el lehet kezdeni az átkapcsolás folyamatát. Az első lépés egy kényszerített logfájl váltás, majd pedig az átkapcsolást megvalósító parancs kiadása standby üzemmódra. Mindez az éles rendszeren (ORADB_P) történik.

```
SQL> alter system switch logfile;
System altered

SQL> alter database commit to switchover to physical standby with
session shutdown;
Database altered
```

A második lépés – ha az előző lépés sikerrel zárult – a jelenlegi standby adatbázis (ORADB_SF) szerepkörének megváltoztatása. Ez csak akkor valósul meg, ha nincs egyetlen felhasználó sem bejelentkezve a standby adatbázisba.

```
SQL> alter database commit to switchover to primary;
Database altered
```

Harmadik lépésként a korábbi elsődleges (ORADB_P) adatbázist újra kell indítani:

```
SQL> shutdown immediate
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.
Database mounted.
```

Negyedik lépésben ugyanezt a folyamatot meg kell megismételni az új elsődleges adatbázison (ORADB_SF), annyi eltéréssel, hogy itt megnyitható az adatbázis.

```
SQL> shutdown immediate
ORACLE instance shut down.
```

```
SQL> startup open
ORACLE instance started.
Database mounted.
```

Ötödik lépésben pedig megnyitható olvasásra, és elindítható a standby üzemmódban, az új másodlagos adatbázis (ORADB_P):

```
SQL> alter database open read only;
Database altered

SQL> alter database recover managed standby database disconnect from session;
System altered
```

Nem maradt más hátra, mint ellenőrizni a folyamat sikerességét, először a szerepeket, majd pedig az archived log fájlok továbbítását:

Korábbi éles rendszer (ORADB_P):

```
SQL> select name, open_mode, protection_level, database_role from v$database;

NAME      OPEN_MODE            PROTECTION_LEVEL     DATABASE_ROLE
--------- -------------------- -------------------- ----------------
ORADB_P   READ ONLY WITH APPLY MAXIMUM PERFORMANCE  PHYSICAL STANDBY
```

Korábbi standby rendszer (ORADB_SF):

```
SQL> select name, open_mode, protection_level, database_role from v$database;

NAME      OPEN_MODE            PROTECTION_LEVEL     DATABASE_ROLE
--------- -------------------- -------------------- ----------------
ORADB_P   READ WRITE           MAXIMUM PERFORMANCE  PRIMARY
```

A szerepek valóban felcserélődtek, megtörtént az átkapcsolás a két oldal között. A logfájlok továbbításának ellenőrzéséhez, ki kell adni a megszokott `alter system switch logfile;` utasítást az új elsődleges rendszeren (ORADB_SF), majd a standby oldalon megvizsgálni, hogy a szolgáltatás helyesen működik-e. Amint az látható a folyamat sikerrel zárult. Az új standby adatbázis (ORADB_P) megkapja a logfájlokat és azokat fel is dolgozza:

```
SQL> select sequence#, applied from v$archived_log order by 1;
```

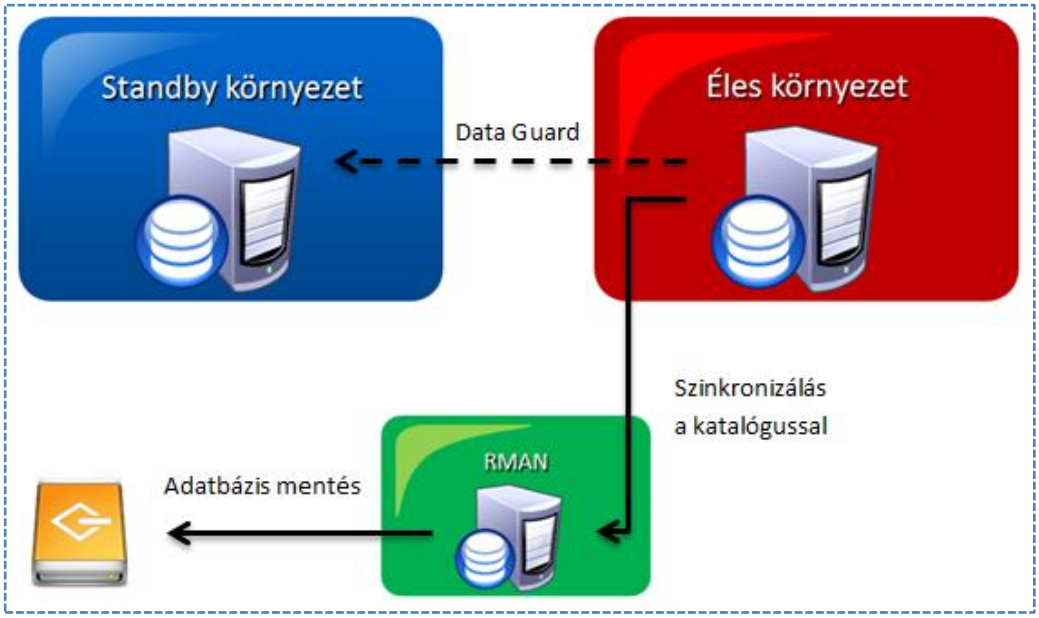
Switch logfile előtt

```
SEQUENCE# APPLIED
--------- ---------
       46 YES
       47 YES
       47 YES
       48 YES
       49 YES
       50 YES
```

Switch logfile után

```
SEQUENCE# APPLIED
--------- ---------
       46 YES
       47 YES
       47 YES
       48 YES
       49 YES
       50 YES
       51 IN-MEMORY
```

Összefoglalásul a teszt sikerrel zárult: az adatbázis szerepek felcserélődtek, és az új standby oldal felé is működik a logok átvitele. Az adatbázis-kezelő rendszer még egy lépéssel közelebb került a kitűzött célhoz. A nem tervezett leállások esetén a failover átkapcsolás is megvalósítható az eddig látottak alapján. A folyamatábrán jól látható ennek a fázisnak az eredménye: a standby oldal az átkapcsolás hatására átvette az elsődleges rendszer szerepét. (Oracle, 2010b)



13. ábra: Ötödik fázis – Átkapcsolási teszt végrehajtása

4.4. Biztonsági beállítások és auditálás

Az eddig bemutatottak nagy része a külső támadásoktól, vagy a felhasználóktól független veszélyek ellen nyújt védelmet. A belső veszélyforrások elleni védelmi mechanizmusok kifejtése következik most. A gyakorlati megoldások bemutatásánál azt a három irányvonalat követem, amelyet az elméleti felvezetésben is bemutattam. Szó

lesz tehát jogosultság-kezelésről, az autentikációról és a felhasználók nyomon követéséről. Ezeknek a szempontoknak a vizsgálatáról gondoskodni kell annak érdekében, nehogy váratlan támadás érje a rendszert, amelynek következtében adatvesztésre, rendszerleállásra kerülne sor vagy illetéktelenek birtokába kerülnének információk. Nem feltétlen kell azonban csak ilyen nagyságrendű dolgokra gondolni, amikor a rendszert tervezzük és paraméterezzük. Az apróbb kiskapuk lezárását is érdemes megvalósítani.

A jogosultság-kezelést illetően érdemes az alapvető Oracle rendszersémák vizsgálatával kezdeni. Ezeknek többségét érdemes zárolni. Arról, hogy milyen sémák léteznek, a következő lekérdezés szolgálhat válaszul:

```
SQL> SELECT 'Előre meghatározott adminisztratív sémák' AS "Séma neve",
       COUNT (*) AS darabszám
  FROM dba_users
 WHERE username IN
          ('ANONYMOUS', 'CTXSYS', 'DBSNMP', 'OLAPSYS', 'OWBSYS',
'ORDPLUGINS', 'ORDSYS', 'OUTLN', 'SYS', 'SYSMAN', 'SYSTEM', 'TSMSYS',
           'WK_TEST', 'WKSYS', 'WKPROXY', 'WMSYS', 'XDB')
UNION
SELECT 'Alapértelmezett minta sémák', COUNT (*) AS darabszám
  FROM dba_users
 WHERE username IN ('BI', 'HR', 'OE', 'PM', 'IX', 'SH')
UNION
SELECT 'Előre meghatározott nem adminisztratív jellegű sémák', COUNT
(*) AS darabszám
  FROM dba_users
 WHERE username IN
          ('APEX_PUBLIC_USER', 'DIP', 'FLOWS_30000', 'FLOWS_FILES',
'MDDATA',
           'ORACLE_OCM', 'PUBLIC', 'SPATIAL_CSW_ADMIN_USER',
           'SPATIAL_WFS_ADMIN_USR', 'XS$NULL');

Séma neve                                           DARABSZÁM
--------------------------------------------------- ----------
Alapértelmezett minta sémák                                  0
Előre meghatározott adminisztratív sémák                     4
Előre meghatározott nem adminisztratív jellegű sémák         2
```

A mintasémák telepítése természetesen teljesen felesleges, de előfordulhat, hogy véletlenül mégis telepítve lettek. Az előre meghatározott sémák általában egy-egy funkcióhoz kapcsolódnak, bizonyos feladatot látnak el. Így például a `dmsys` sémát az adatbányászatért felelős modul használja, az `olapsys` pedig az OLAP modul sémája. Amennyiben ezeket a funkciók nem települnek fel, akkor a sémák sem jönnek létre. Ezeknek a felülvizsgálatáról érdemes gondoskodni. (*Alapati, 2005*)

A magas jogosultsággal rendelkező felhasználókat is érdemes ellenőrizni, illetve a jogokat visszavonni, ha az indokolt. A DBA jogkörrel rendelkező felhasználókat az alábbi lekérdezés adja vissza:

```
SQL> SELECT GRANTEE FROM DBA_ROLE_PRIVS WHERE GRANTED_ROLE='DBA'

GRANTEE
------------------------------
SYS
SYSTEM
DBA_ADMIN
```

Amennyiben olyan felhasználó szerepel itt a listában, aki adott esetben már nem dolgozik az adott vállalatnál, vagy beazonosíthatatlan, akkor érdemes zárolni, majd törölni ezeket a sémákat. Ha már a dba jogkörrel rendelkezők vannak vizsgálat alatt, akkor mindenképp érdemes a password file-ban szereplő jogosult felhasználókat is ellenőrizni. Ebben a következő utasítás nyújthat segítséget:

```
SQL> select * from v$pwfile_users;

USERNAME                       SYSDB SYSOP SYSAS
------------------------------ ----- ----- -----
SYS                            TRUE  TRUE  FALSE
```

Általában ezt a jogot a SYS és SYSTEM felhasználó szokta megkapni. Emiatt ezeknek a felhasználóknak az átlagostól is sokkal összetettebb jelszót kell adni, ami legalább 15 karakter hosszú, kis és nagybetűket, valamint számot és egyéb speciális karaktert is tartalmaz.

Az adatszótár védelme az újabb verziójú Oracle adatbázis-kezelő rendszerben már alapértelmezett. Erről az `o7_dictionary_accessibility=false` paraméter gondoskodik. Érdemes meggyőződni arról, hogy ez a paraméter lehetőleg sohase vegyen fel `true` értéket. Mivel az egyes objektumokat az Oracle adatbázis-kezelő rendszer esetén sémák tartalmazzák – ellentétben az MSSQL-es adatbázis-kezelő rendszerrel, ahol az objektumok az adatbázishoz tartoznak – megfontolandó, hogy minden alkalmazás sémához egy olyan felhasználót kelljen létrehozni, amin keresztül el lehet érni ezeket az objektumokat (*Oracle, 2010a*). Így biztosítva azt, hogy az üzletmenet folyamán ne az alkalmazássémát használják, aminek jogosultsága van akár eldobni táblákat, törölni objektumokat. A technikai sémának pedig csak ki kell osztani a szükséges jogosultságokat, amelyek feltétlenül elengedhetetlenek, valamint alternatív hivatkozásokat – szinonimákat – kell létrehozni az alkalmazásséma objektumaira. Ezzel

párhuzamosan egyénileg definiált szerepköröket is célszerű létrehozni az adatbázisban, amelyek kioszthatók az egyes felhasználók számára. Példának okán ilyen szerepkör lehet a readonly szerepkör is, amelyet a technikai sémákon kívül a felhasználók is megkaphatnak annak érdekében, hogy láthassák az egyes alkalmazássémák tábláinak adatait. Ennek azonban számtalan variációja lehet, ezek segítségével biztosítható az az elv, hogy minden felhasználó csak azokhoz az információkhoz férjen hozzá, amelyek munkavégzésükhöz feltétlen szükséges. Ez a nézőpont alkalmazható a technikai sémákra is.

Ugyanez a felfogás alkalmazható a felhasználói – legyen az technikai vagy rendes értelemben vett felhasználói séma – profilokra is. A profilbeállításokkal is teljes körűen konfigurálható és testre szabható csoportok hozhatók létre aszerint, hogy ki milyen széles jogkörökkel rendelkezhet. A profilokban beállítható fontosabb paraméterek a következők:

Paraméter	Leírás
session per user	Az egyidejű bejelentkezések számát határozza meg felhasználónként
cpu per session	A maximális cpu idő felhasználást jelenti bejelentkezésenként
connect time	Az engedélyezett kapcsolódási idő percben megadva
idle time	A megengedett inaktivitási időtartam percben megadva
logical reads per session	A maximálisan engedélyezett blokkolvasás utasításonként
private sga	A maximális egyedi memória felhasználást határozza meg az SGA-ban. Értéke byte-ban értendő

(*Forrás: Oracle, 2010a*)

Ezeknek a paramétereknek a felhasználásával jól testreszabható profilok hozhatók létre az egyes felhasználói csoportoknak megfelelően. Ezek képezhetik az alapját a felhasználók korlátozásának is. Egyes csoportok kaphatnak teljes jogkört, korlátozások nélkül, például az adatbázis-adminisztrátorok, de egyes csoportok, például a fejlesztők, vagy a teljesen általános felhasználók már korlátozásokkal láthatóak el.

A profiloknál már szóba kerültek a jelszavakkal kapcsolatos beállítások, így érdemes áttérni az autentikáció kérdéskörére. A profilbeállításoknál általánosan a következő beállítások érhetőek el:

Paraméter	Leírás
failed login attempts	A sikertelen bejelentkezések száma mielőtt a felhasználó zárolódik
password lifetime	A jelszó használati ideje napokban
password reuse time	A jelszó újrahasználhatóságáig szükséges napok száma
password reuse max	Meghatározza, hogy egy jelszó újrafelhasználásáig hányszor kell új jelszót használni
password lock	A zárolás időtartamát adja meg napokban, ha elérik a meghatározott failed login attempts-ben meghatározott értéket
password grace time	A jelszó lejárat előtti figyelmeztetési időszakot határozza meg
password verify function	A jelszavakkal kapcsolatos elvárásokat beállító funkció.

(*Forrás: Oracle, 2010a*)

Ezek a beállítások a profilokkal együtt állíthatóak be, de érdemes mindenhol ugyanazt a beállítást alkalmazni. Célszerű minél szigorúbb beállításokat használni. Példaként a követező profil létrehozása javasolt például az adatbázis-adminisztrátorok számára:

```
SQL> create profile dba limit
      failed_login_attempts 3
      password_life_time 30
      password_reuse_max 10
      password_lock_time unlimited
      password_grace_time 3
      password_verify_function verify_function_11g;
```

A dba profil esetén három hibás jelszó megadása után a felhasználói séma zárolódik, mindaddig, amíg egy másik dba jogosultságú felhasználó fel nem oldja. 30 naponként meg kell változtatni a jelszót. Egy jelszó 10 eltérő használata után használható újra. A jelszó erősségére vonatkozó kritériumok vizsgálatára szolgál az utolsó paraméter. A verify function egy beépített Oracle megoldás, amelyet ajánlott is telepíteni és használni. Ez hivatott a jelszavak komplexitásának vizsgálatára és ellenőrzésére. Ennek telepítésekor – amely az alábbi módon lehetséges – két függvény jön létre, és módosul az alapértelmezett profile is:

```
SQL>@$ORACLE_HOME\RDBMS\ADMIN\utlpwdmg.sql
Function created
Profile altered
Function created
```

A verify function módosítható saját szempont szerint, csupán a `SYS.VERIFY_FUNCTION_11G` függvényt kell módosítani. A szükséges paraméterek

megtarthatók, a többi kivehető, minden csak beállítások kérdése. Követelmények határozhatók meg a következő paraméterek mindegyikére: a jelszó hossza, a jelszó bonyolultsága, a jelszó nem hasonlít-e túlságosan az előző jelszóra. Néhány egyénileg kialakított példa a függvényből:

```
-- A jelszó minimális hosszának ellenőrzése
IF length(password) < 10 THEN
raise_application_error(-20001, 'A jelszó hossza rövidebb, mint 10
karakter!');
   END IF;

-- A jelszó és felhasználónév egyezésének vizsgálata

FOR i in REVERSE 1..length(username) LOOP
reverse_user := reverse_user || substr(username, i, 1);
END LOOP;
IF NLS_LOWER(password) = NLS_LOWER(reverse_user) THEN
raise_application_error(-20003, A jelszó megegyezik a
felhasználónévvel');
END IF;
```

A verify function segítségével széles körűen testreszabható a jelszavakkal és az autentikációval kapcsolatos beállítások, így megelőzve a lehetséges felhasználói visszaéléseket, amelyekből komolyabb problémák is származhatnak. Célszerű mindenképpen így biztosítani az autentikációt és nem pedig az operációs rendszer autentikációját választani az adatbázishoz hozzáféréshez.

Az eddig bemutatott megoldások után a felhasználók tevékenységének monitorozására is érdemes kitérni. Az adatbázis adminisztrátoroknak lehetősége van az összes cselekvés auditálására, ami az adatbázisban történik. Azonban minden cselekedet rögzítése nem optimális megoldást, hiszen akkor van értelme az auditálásnak, ha az nem jár jelentős teljesítménybeli vonzattal. Minden cselekedet rögzítése erőforrásokat köt le, von el az üzleti alkalmazásoktól. Fontos tehát csak azt auditálni, amit később felhasznál az auditor, amiből jelentések készülnek, és aminek üzleti jelentősége van. Két főbb részterületre térek ki röviden ebben az alpontban: az egyéni megoldásokra, és az általános adatbázis auditra (*Oracle, 2010a*).

Az egyéni megoldások számos lehetőséget adnak az adatbázis-adminisztrátorok számára. A triggerek, PL/SQL megoldások biztosítják, hogy tényleg azokat az információkat tárolja el az adatbázis a felhasználókról, amelyek lényegesek. Egy egyszerű példa ennek bemutatására a következő koncepció, amelyet alkalmazok a felhasználói be, illetve kilépések auditálására:

Az első lépés egy adatokat tartalmazó tábla kialakítása:

```
-- Audit információkat tartalmazó tábla létrehozása
CREATE TABLE
   dba_admin.user_audit_table
(
   user_id           VARCHAR2(30),
   session_id           NUMBER(8),
   HOST              VARCHAR2(30),
   logon_day                DATE,
   logoff_day               DATE,
   elapsed_minutes       NUMBER(8)
);
```

Ebben a táblában tárolom majd az adatokat a dba_admin séma alatt. A felhasználónév, a kiszolgáló neve, a bejelentkezési idő és a kijelentkezési idő mellett az eltelt időtartam is rögzítésre kerül. Ezután létrehozhatóak a szükséges triggerek, amelyek feltöltik adatokkal ezt a táblát.

```
-- Bejelentkezés után lefutó trigger
CREATE OR REPLACE TRIGGER logon_audit_trigger
   AFTER LOGON ON DATABASE
BEGIN
   -- Dátumformátum beállítása
   EXECUTE IMMEDIATE 'ALTER SESSION SET NLS_DATE_FORMAT=''YYYY-MM-DD
HH24:MI:SS''';

   -- Adatok beszúrása az audit táblába
   INSERT INTO dba_admin.user_audit_table
        VALUES (USER, SYS_CONTEXT ('USERENV', 'SESSIONID'),
                SYS_CONTEXT ('USERENV', 'HOST'), SYSDATE, NULL, NULL);
END;
/

-- Kijelentkezés előtt lefutó trigger
CREATE OR REPLACE TRIGGER logoff_audit_trigger
   BEFORE LOGOFF ON DATABASE
BEGIN
   UPDATE dba_admin.user_audit_table
      SET logoff_day = SYSDATE
    WHERE SYS_CONTEXT ('USERENV', 'SESSIONID') = session_id;

   UPDATE dba_admin.user_audit_table
      SET elapsed_minutes = ROUND ((logoff_day - logon_day) * 1440)
    WHERE SYS_CONTEXT ('USERENV', 'SESSIONID') = session_id;
END;
/
```

Az eredményt pedig az alábbi lekérdezéssel ellenőrizhetjük:

```
SQL> select * from dba_admin.user_audit_table;

USER_ID    HOST     LOGON_DAY            LOGOFF_DAY           ELAPSED_MINUTES
---------- -------- -------------------- -------------------- ---------------
DBA_ADMIN  SERVER1  2011-04-17 11:15:32   2011-04-17 11:21:00                5
```

Természetesen ez csak egy lehetőség a sok közül, a megoldások száma szinte végtelen. Amennyiben viszont az Oracle adatbázis-kezelő rendszer megoldásaira szeretnénk hagyatkozni akkor érdemes az általános adatbázis auditot (standard database audit) választani. Erről már a korábbiakban is volt szó, ennek engedélyezéséhez a következő adatbázis paramétert kell beállítanunk, mint azt már korábban is láthattuk:

```
SQL> alter system set audit_trail = db scope=spfile;
System altered
```

Ebben az esetben az audit információk a `SYS.AUD$` táblában kerülnek rögzítésre. Alapesetben ehhez a táblához csak a megfelelő privilégiummal rendelkező felhasználók férhetnek hozzá. Az „audit" utasítás segítségével monitorozható a felhasználók tevékenysége, teljesen személyre szabhatóan. Auditálhatóak az általános adatbázis audit segítségével a DDL műveletek (pl. create table, alter table), a DML műveletek (insert, update, delete), a lekérdezések, és egyéb események is. Az Oracle automatikusan számos információt tárol el abban az esetben, ha az audit beállítások úgy kívánják. Az információ vagy közvetlenül a `SYS.AUD$` táblából nyerhető, vagy az audit nézeteken keresztül, mint például a `DBA_STMT_AUDIT_OPTS`, `DBA_OBJ_AUDIT_OPTS`, vagy `DBA_AUDIT_TRAIL`. A következő utasítások szolgálnak példával az általános adatbázis auditra. Ezek az utasítások csupán példák, egy kialakított üzleti igényeket kiszolgáló környezetben természetesen sokkal nagyobb fokú és megtervezettebb audit beállításokat kell kialakítani.

```
-- A sikertelen bejelentkezések auditálása
SQL> audit session whenever not successful;

-- A felhasználó összes tevékenységének auditálása
SQL> audit all by dba_admin by access;

-- A felhasználó összes lekérdezésének auditálása
SQL> audit select any table by dba_admin by access;

-- Adott táblán a felhasználói cselekvések auditálása
SQL> audit insert, update on dba_admin.user_audit_table by access;
Audit complete
```

A `SYS.AUD$` táblának a táblatere alapértelmezetten a SYSTEM táblatér. Sok felhasználó esetén és részletes audit beállítások mellett a keletkezett audit információk elég nagy helyet foglalhatnak el, és fennáll a veszélye annak, hogy a táblatér betelik, ami veszélyt jelenthet az adatbázis működésére nézve. Ennek elkerülése érdekében az egyik

megoldás az, ha időszakosan mentés készül a tábláról, majd ezt követően törlésre kerül a tartalma, vagy pedig saját táblateret rendelünk hozzá. Célszerű ezt a megoldást választani. A táblatér létrehozása a korábban látottak alapján a következő:

```
CREATE TABLESPACE audit_tbs
  DATAFILE 'c:\oracle\oradata\oradb_p\audit01.dbf'
  SIZE 50M AUTOEXTEND ON NEXT 2M;
```

Tablespace created

Az `AUD$` tábla alapértelmezett táblaterének megváltoztatása a következő utasítással valósítható meg:

```
BEGIN
   dbms_audit_mgmt.set_audit_trail_location
       (audit_trail_type => dbms_audit_mgmt.audit_trail_db_std,
        audit_trail_location_value => 'audit_tbs'
        );
END;
/
```

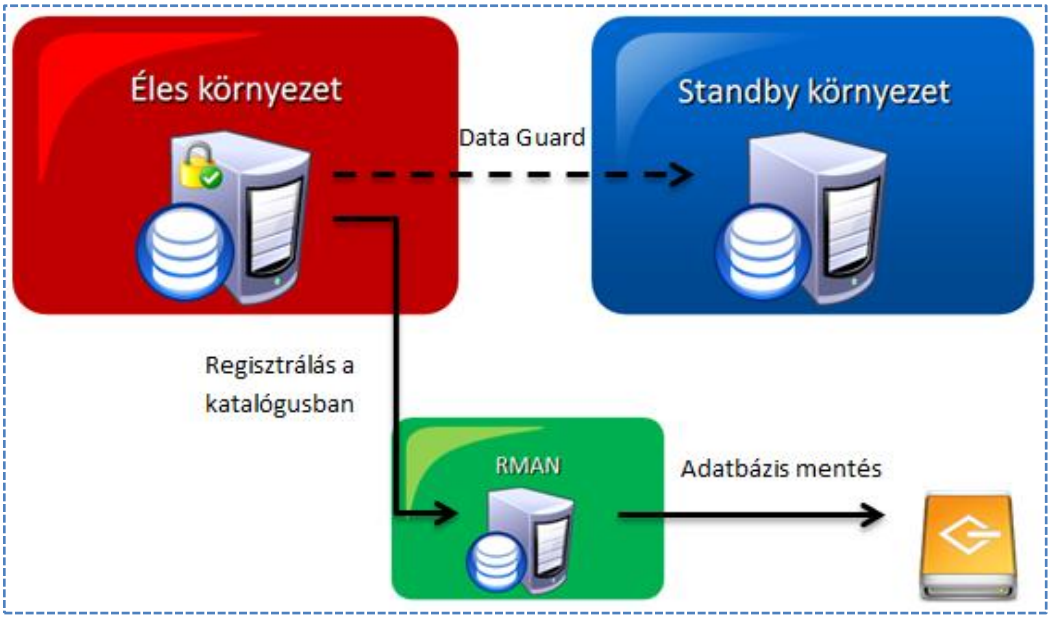
PL/SQL procedure successfully completed.

Végezetül pedig ellenőrizhető az eredmény egy lekérdezés segítségével.

```
SQL> select table_name, tablespace_name from dba_tables where
table_name = 'AUD$';

TABLE_NAME           TABLESPACE_NAME
-------------------- ----------------------
AUD$                 AUDIT_TBS
```

Ezek az audit beállítások szolgálnak példaként egy-egy vállalat számára az adatbázis-kezelő rendszert illetően. Ezekre a beállításokra lehet alapozni az auditjelentéseket, és a szükséges problémákat megelőzendő lépéseket. Az alkalmazások egyedi auditálási megoldásai mellett, ezeket az Oracle által nyújtott audit szolgáltatásokat ajánlott igénybe venni, így eleget téve az üzletfolytonosság-menedzsment által szabott igényeknek. A rendszer egyúttal ennek a fejezetnek a végére elérte végső állapotát, azt, amit az 5. fejezetben felvázoltam. Kialakult egy magas rendelkezésre állású, hibatűrő adatbázis-kezelő rendszer környezet. Ez látható ismét a következő ábrán:



14. ábra: Hatodik fázis - Biztonsági beállítások érvényesítése

5. Összefoglalás

100%-os rendelkezésre állású informatikai rendszert – dolgozatom szempontjából tekintve adatbázis-kezelő rendszert – nem egyszerű feladat megvalósítani. Számos veszélyforrást kell kezelni, és egyúttal az üzleti igényeket is figyelembe kell venni. Amint látható dolgozatomból az Oracle biztosítja azokat a szoftveres megoldásokat – ma már akár hardveres megoldásokat is – amelyek segítségével megvalósítható egy hibatűrő, magas rendelkezésre állású adatbázis-kezelő rendszer. Azonban ezeket a megoldásokat a megfelelő módon kell használni annak érdekében, hogy a kívánt cél elérhetővé váljon, valamint tudni kell azt is, hogy milyen esetekben milyen megoldást kell választani. Az informatikai projektek sikerét nagymértékben meghatározza a tervezési fázis minősége és sikeressége. Nincs ez másként egy adatbázis-kezelő rendszer kialakításánál sem. Előre meg kell tervezni, fel kell mérni a rendszer funkcionalitását és paramétereit. Ennek megfelelően kell meghatározni, milyen mértékű védelmi mechanizmusokat kell alkalmazni és megvalósítani. A magas védelem és rendelkezésre állás fontos a vállalat szempontjából, azonban ha ennek költségei meghaladják a várható veszteségeket egy leállás esetén, akkor érdemes újragondolni szükség van-e ilyen mértékű védelemre.

Dolgozatomban bemutattam az üzletfolytonosság-menedzsment alapjait, kialakítását és a különböző ehhez kapcsolódó egyes dokumentumokat. Bemutattam a különböző Oracle megoldásokat, amelyek hozzájárulnak az adatbázis-kezelő rendszer magas rendelkezésre állásához. Láthatóvá vált miként véd a hardveres meghibásodásoktól az Oracle Real Application Cluster, hogyan biztosítja az Active Data Guard a standby adatbázisok folyamatos frissítését. A Recovery Manager keretében pedig bemutattam a legfontosabb mentési módokat, amelyeket egy adatbázis-adminisztrátornak ismernie és használnia kell. Ezeken a szolgáltatásokon kívül természetesen több egyéb megoldás létezik, amely alkalmazható, legyen az Oracle vagy más vállalat terméke. Az általam bemutatottak egy közepes méretű vállalat esetén elegendőek kell, hogy legyenek. Nem tértem ki olyan érdekes lehetőségekre, minta a Real Application Testing, az Oracle Flashback Technology, az Oracle Audit Vault vagy például az Active Data Guard witness szerver megoldása. Ezek szintén hasznos szolgáltatások, amelyek alkalmazása nagy vállalatok esetén megfontolandó.

A gyakorlati részben bemutattam egy középvállalat számára megfelelő adatbázis-kezelő rendszer környezet kialakítását. Látható volt, hogy milyen fontos szereppel bír a rendszer megtervezése és a kezdeti paraméterek beállítása. Lényeges, hogy ebben a szakaszban még könnyedén változtathatók a rendszer beállításai, az éles működés megkezdése után azonban a paraméterek állítása már legtöbb esetben az adatbázis újraindítását igényli. Ebből látható, hogy a tervezési fázis az informatikai rendszerek bevezetésekor kiemelt fontossággal bír. Bemutattam egy-két egyszerű, de mindenképp hasznos megoldást annak érdekében, hogy az adatbázis működését hatékonyabbá tegyem. Első lépésként ezután bemutattam a Recovery Manager legfontosabb funkcióit a gyakorlatban. A konfigurálás után az adatbázisról és a hozzá tartozó fájlokról mentést is készítettem. A folyamat másik részét is elvégeztem: helyreállítottam az adatbázist a meglévő mentés alapján. A Recovery Manager az egyik legfontosabb védelmi megoldás a standby adatbázisok mellett, így használata nélkülözhetetlen. Ha minden egyéb megoldás kudarcot vall, akkor az RMAN-al még mindig helyreállítható az adatbázis a katasztrófa után. Mindezek után konfiguráltam az elsődleges adatbázist abból a célból, hogy alkalmas legyen a standby környezet kialakítására. A paraméter beállításokra érdemes nagy figyelmet szentelni minden esetben, mivel ezek határozzák meg az adatbázis működését és helytelen beállítások esetén az adatbázis leállását is okozhatják. A standby adatbázis kialakításának kétféle módját is ismertettem: első esetben manuálisan, majd pedig a Recovery Manager használatával. Az eljárás mindkét esetben hasonló: a paraméter beállítások után konfigurálni kell a listener-t, Windows szervizt és password file-t kell készíteni, ki kell alakítani a megfelelő könyvtárszerkezetet. Ezek után indítható el a standby adatbázis és a tükrözés folyamata. A folyamat eredményét a switchover teszttel vizsgáltam. Ennek keretében a két adatbázis szerepkörét felcseréltem, és a standby adatbázis átvette az éles szerepét. A standby adatbázis jelenti a legfontosabb védelmet a leállások ellen. Véleményem szerint az RMAN mellett ez a szolgáltatás – a Data Guard – az, amelyik minden vállalat számára szükséges ahol komolyabb adatbázisok üzemelnek. A Data Guard biztosítja, hogy az éles és standby adatbázis szinkronban legyen. A biztonsági megoldásokat taglaló részben ismertettem azokat a lehetőségeket, amelyeket minden adatbázis adminisztrátornak célszerű ismernie. A profil és jogosultság beállítások mellett kiértem röviden az jelszókezelésre és a

felhasználók auditálására is. A fejezetben leírtak kellő védelemmel szolgálnak a belső, illetve a felhasználói veszélyforrások ellen. Ugyanis a vállalaton belülről érkező fenyegetések ugyanolyan súlyosak lehetnek, mint a külső veszélyek. A bizalmas adatok védelmének biztosításáról gondoskodni kell.

Összefoglalva tehát véleményem szerint a dolgozatomban bemutatottak elegendő biztonságot kell, hogy nyújtsanak egy közepes méretű vállalat adatbázis-kezelő rendszere védelmének garantálásához. Ezáltal biztosítva az üzletmenet-folytonosságot, valamint a magas rendelkezésre állást. Az informatikai felsővezetőknek nem kell felhasználói szinten ismernie és kezelnie ezeket a szolgáltatásokat, de tudnia kell, hogy mire szolgálnak ezek, és hogyan tudják a vállalati működést segíteni. Így megfelelő döntést tudnak hozni az üzletfolytonosságot érintő kérdésekben. A bemutatott gyakorlati megoldások hivatottak minimalizálni a tervezett és nem tervezett leállások során kiesett időt. Egyúttal a BCP és DRP tervekben is fontos szerepet kell, hogy kapjanak ezek a koncepciók. Az Oracle által nyújtott adatbázis-kezelő rendszerekhez kapcsolódó megoldások képesek lefedni az üzletfolytonosság menedzsment minden részterületét, és bátran lehet építeni rájuk, ha adatbázis-rendszereink rendelkezésre állásáról van szó.

6. Irodalomjegyzék

6.1. Írott szakirodalom:

Kevin Loney (2006): *Oracle Database 10g – Teljes referencia, Panem Kiadó*

Matthew Hart, Scott Jesse (2004): *Oracle Database 10g: High Availablity with RAC Flashback & Data Guard*

Raffai Mária (1999): *BCP: Üzletmenet-folytonosság biztosítása: Megelőzési, felkészülési és helyreállítási terv, Novadat Kiadó*

Ron Ben Natan (2009): *HOWTO Secure and Audit Oracle 10g and 11g*

Sam R. Alapati (2005): *Expert Oracle Database 10g Administration*

6.2. Internetes szakirodalom:

IDC (2008): The Diverse and Exploding Digital Universe – letöltve: 2011.04.23
http://www.emc.com/collateral/analyst-reports/diverse-exploding-digital-universe.pdf

Oracle (2005): Oracle® Database Concepts 10g Release 2 (10.2) – letöltve: 2011.04.22
http://download.oracle.com/docs/cd/B19306_01/server.102/b14220.pdf

[a] Oracle (2010): Oracle® Database 2 Day + Security Guide 11*g* Release 2 (11.2) – letöltve: 2011.04.22
http://www.oracle.com/pls/db112/to_pdf?pathname=server.112/e10575.pdf

[b] Oracle (2010): Oracle® Database High Availability Overview 11*g* Release 2 (11.2) – letöltve: 2011.04.22
http://www.oracle.com/pls/db112/to_pdf?pathname=server.112/e17157.pdf

[c] Oracle (2010): Oracle® Data Guard Concepts and Administration 11*g* Release 2 (11.2) – letöltve: 2011.04.22
http://www.oracle.com/pls/db112/to_pdf?pathname=server.112/e17022.pdf

[d] Oracle (2010): Oracle® Database Backup and Recovery Reference 11*g* Release 2 (11.2) – letöltve: 2011.04.22
http://www.oracle.com/pls/db112/to_pdf?pathname=backup.112/e10643.pdf

[e] Oracle (2010): Oracle® Database Backup and Recovery User's Guide 11*g* Release 2 (11.2) – letöltve: 2011.04.22
http://www.oracle.com/pls/db112/to_pdf?pathname=backup.112/e10642.pdf

Ponemon Institute (2010) - 2009 Annual Study: Cost of Data Breach – letöltve: 2011.04.22
http://www.ponemon.org/local/upload/fckjail/generalcontent/18/file/US_Ponemon_CODB_09_012209_sec.pdf

6.3. Ábrajegyzék

6.4. Táblázatok