# ÖNÁLLÓ LABOR
# VII. SZEMESZTER

**Sümeghy Tamás Pál**
**GFHSRE**

# ORACLE SPATIAL / BUSINESS INTELLIGENCE

## A technológiák általános bemutatása

### Oracle Locator / Spatial

A Locator az Oracle 8i újdonsága (Standard és Enterprise Edition). Olyan helyfüggő és helymeghatározó alap funkció, amely a legtöbb felhasználói alkalmazáshoz szükséges. Ez az alap funkció az, hogy különböző térinformációkat (pl. szélességi és hosszúsági koordináták) tudunk az adatainkkal együtt, egy helyen tárolni. Ezen kívül, plusz szolgáltatásként ezen kiegészítő adatok analízisére is lehetőséget ad.

A Spatial szintén az Oracle 8i-ben jelent meg (Enterprise Edition), nagyban hasonlít a Locator-re, az alapjaik azonosak. Ugyanazt az alapvető objektumtípust használják (SDO_GEOMETRY), azonos metaadatok és indexelési sémák jellemzőek rájuk. Tulajdonképpen a Locator tartalmazza a Spatial alapfunkciókat, de a fejlesztések mind a Spatial-ben kerültek implementálásra.

Míg a Locator alapfunkciókat biztosít csak térbeli adatok analíziséhez (pl. képes megtalálni mindazon adatokat, amelyek topológiai kapcsolatban vannak más adatokkal), addig a Spatial erre alapozva egy összetettebb szolgáltatást is képes nyújtani:

- képek és rácshálón értelmezett adatok, metaadatok tárolása és kezelése
- hálózatok és topologikus adatmodellek létrehozása és analízise
- szöveg alapú címinformációk hosszúsági/szélességi adatokká konvertálása geokódolás segítségével
- irányítási információk szolgáltatása egy beépített útvonalkereső motor segítségével (gyakorlatilag a mai GPS berendezések szolgáltatása)
- többdimenziós térbeli analízis és adatbányászat térinformációs adatok között

### Spatial

A Spatial lehetséges definíciói:

- egy séma, ami geometriai adattípusok tárolását, szintaxisát és szemantikáját írja le
- egy térbeli indexelési mechanizmus
- operátorok, funkciók és eljárások gyűjteménye, amelyek segítségével térben összekapcsolt adatbázisokat kérdezhetünk le vagy térinformációt hordozó adatokon végezhetünk analízist
- topologikus adatmodell, hogy az adatra úgy tekinthessünk, mint csúcspontra, élre vagy felületre egy topológiában
- hálózati adatmodell, melynek segítségével tulajdonságokat vagy objektumokat modellezhetünk, mint csomópontok és a köztük lévő kapcsolatok egy hálózatban
- GeoRaster adatformátum, ami egy raszterkép és egy rácsháló, valamint a kettőt összekapcsoló metaadatok összessége

A Spatial egy objektum-kapcsolat modellel reprezentálja a különböző geometriai alakzatokat. Erre egy natív térbeli adattípust használ, ami alapvetően vektor jellegű, ez az SDO_GEOMETRY. Egy Oracle tábla egy vagy több ilyen típusú oszlopot tartalmazhat. Ez az objektum-kapcsolat modell az Open GIS ODBC/SQL specifikációban jelenik meg, mint geometriai típusokkal kiegészített SQL lekérdező nyelv. Ez a megközelítés számos előnnyel jár, például:

- sokféle geometriai alakzatot támogat
- könnyen kezelhető indexek létrehozásakor és karbantartásakor valamint lekérdezések optimalizálásánál
- ugyanúgy lehet kezelni térbeli adatokat, mint az adatbázis többi adatát, sőt kombinált lekérdezéseket is írhatunk üzleti és térbeli információkra

## Spatial alapvető geometriai elemei

Egy geometriai alakzat egyenes szakaszok és görbedarabok rendezett sorrendjéből áll össze. Ezt a sorrendet a geometriai alakzat típusa határozza meg. A Spatial sok különféle alaptípust definiál, a bonyolultabb alakzatok ezek kombinációjából jöhet létre. Az alaptípusok két dimenzióban:

- pontok és ponthalmazok
- egyenes szakaszok, ezekből képzett töröttvonalak
- poligonok (zárt töröttvonalak)
- körívek, ezekből képzett töröttvonalak
- körívekből álló poligonok
- körök
- optimalizált téglalapok

(Megj.: önmagát metsző poligon nem létezik, önmagát metsző töröttvonal viszont igen)

Ezeket az alakzatokat kétdimenziós koordináta rendszerben értelmezzük, így pl. egy pontot két koordinátával (x és y, vagy hosszúsági és szélességi adatok) adhatunk meg.

Ezek a megállapítások a kétdimenziós esetben voltak érvényesek. A Spatial támogatja a több dimenziós (3 vagy 4) alakzatok tárolását, viszont a legtöbb funkció csak az első két koordinátát használja, míg az operátorok nagy része nem engedélyezett, ha az indexelés három vagy több dimenziós adatokon történt meg.

## Az adatmodell

A Spatial adatmodellje egy hierarchikus struktúra, amely alapelemekből, geometriai alakzatokból és rétegekből épül fel. A rétegek geometriai alakzatokból állnak, míg ezeket alapelemek alkotják.

Az elemek az alapvető építőkockák, ezek a pont, a szakasz és a töröttvonal, valamint a poligon. A pontot egy koordinátapárral, egy szakaszt két végpontjának koordinátapárjával adunk meg, egy töröttvonal esetében minden törési pontra meg kell adnunk ezeket az adatokat.

A geometriai alakzatok az alapelemek egy rendezett halmazaként definiálhatóak. Állhatnak mindössze egyetlen alapelemből, de alapelemek homogén vagy heterogén halmazából is.

Egy réteg olyan geometriai alakzatok gyűjteménye, melyeknek az attribútumhalmaza azonos. Például egy térinformatikai rendszerben egy réteg tartalmazza a topográfiai információkat, egy másik a népsűrűség adatait, míg egy harmadik az utak és hidak hálózatát.

## Koordinátarendszerek

Bármely térbeli információnak csak akkor van értelme, ha valamilyen viszonyítási rendszerhez vagy koordinátarendszerhez képest tudjuk megadni a koordinátáit. A viszonyítás alapja lehet a Föld (mint például a szélességi és hosszúsági adatok esetén), de lehet a Földtől független is.

A Spatial 8.1.6-os verziója előtt az adatokhoz nem rendeltek külön saját koordinátarendszert. Minden számításhoz a Descartes-féle koordinátarendszert használták, ez azonban pontatlanságokat okozott például szélességi és hosszúsági adatok számításánál. A 8.1.6-os verziótól kezdve négyféle koordinátarendszer közül választhatunk, ezek között tetszőleges átjárás biztosított.

- Descartes koordinátarendszer: egy meghatározott pontból (az origóból) kiinduló két (vagy három) merőleges egyenes által megadott viszonyítási alap
- geográfiai koordináták: tulajdonképpen a szélességi és hosszúsági adatok, melyek nagyban hasonlítanak a matematikai polárkoordinátákra
- projektív koordináták: síkbeli Descartes-féle koordinátarendszer, amit úgy kaphatunk, hogy egy földfelszíni pontból a matematikai vetítés műveletével leképezzük a koordinátarendszert egy síkra
- helyi koordináták: olyan Descartes-féle koordinátarendszer, amely független a Földtől

## Lekérdezés

A Spatial kétszintű lekérdező modellt alkalmaz a térbeli lekérdezések és összekapcsolások feloldásához. Ez annyit jelent, hogy két elkülönülő operátor szükséges a feloldáshoz. A konkrét eredményhalmaz a két operátor kombinált végrehajtása utáni kimeneti halmaz. A két operátort elsődleges és másodlagos szűrőnek nevezzük.

Az elsődleges szűrő gyorsan és egyszerűen kiválasztja azokat a potenciális rekordokat, amiket a másodlagos szűrő felé továbbításra érdemesnek tart. Egy közelítést alkalmaz a geometriai alakzatokra, és ezeket a közelített adatokat hasonlítja össze, ezzel is gyorsítva a futást és csökkentve a költségeket. Az elsődleges szűrő használata azt eredményezi, hogy egy lekérdezés teljesítménye nem függ az adathalmaz méretétől, mivel egy adott lekérdezés esetén az elsődleges szűrő kimenete mindig ugyanakkora méretű származtatott adathalmaz. A másodlagos szűrő aztán már pontos számítások alapján hozza létre az eredményhalmazt, pontos választ adva a lekérdezésre. Ezek számításigényes feladatok, de az elsődleges szűrő miatt már egy relatíve szűk halmazzal kell csak dolgoznunk, nem a teljes adatmennyiséggel. Nem minden esetben szükséges mind a két szűrő használata, bizonyos esetekben elegendő az elsődleges szűrő is.

Az elsődleges szűrő megvalósításához szükséges az adatok térbeli indexelése. Mivel az elsődleges szűrőnek igen gyorsan és hatékonyan kell az adatok egy részhalmazát kiválasztania, ez az igény meghatározza az indexelés karakterisztikáját.

## Indexelés

A Spatial egy igen erőteljes és gyors indexelést használ, amit R-fának hívnak. Mint minden indexnek, egy térbeli indexnek is egy olyan mechanizmust kell adni, ami gyorsítja és korlátossá teszi a keresést. Ebben az esetben ez a mechanizmus térbeli kritériumokon alapszik.

Egy térbeli indexnek két dolgot kell tudnia. Egyrészt meg kell találnia olyan indexelt adatmezővel rendelkező objektumokat, melyek kapcsolatban vannak egy megadott ponttal vagy területtel (keresés, lekérdezés), másrészt olyan objektumpárokat kell találnia, melyekben van indexelt adatmező és térbeli kapcsolatban vannak egymással (térbeli összekapcsolás).

## Adatszerkezet

A Spatial tartalmaz egy objektum adattípus-halmazt, ezeket az objektum adattípusokat felhasználó operátorokat, függvényeket és eljárásokat. Egy objektumot egy egyszerű sorban tudunk tárolni, az oszlop típusa pedig *SDO_GEOMETRY*. Ezekről az objektumokról lesz szó az alábbiakban.

Az *SDO_GEOMETRY*-t így definiálták:

```
CREATE TYPE sdo_geometry AS OBJECT (
 SDO_GTYPE NUMBER,
 SDO_SRID NUMBER,
 SDO_POINT SDO_POINT_TYPE,
 SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
 SDO_ORDINATES SDO_ORDINATE_ARRAY);
```

Az egyes részek jelentése:

*SDO_GTYPE*: az alakzat típusát adja meg. Egy négyjegyű szám, melynek formátuma *dltt*. Az első számjegy (d) adja meg a dimenziószámot, ez lehet 2, 3 vagy 4. Magasabb dimenziók esetén használhatunk lineáris viszonyítási rendszert (LRS), a második számjegy (l) adja meg, hogy melyik dimenzió adja meg az ehhez használt mértéket. Ha nem használunk LRS-t vagy alapértelmezetten az utolsó dimenziót akarjuk megadni, akkor ez az érték 0. Az utolsó két számjegy (*tt*) az alakzatot írja le, ez jelenleg 00-tól 07-ig használt, a többi érték későbbi felhasználásra van.

Példa: *SDO_GTYPE = 2003*, tehát egy kétdimenziós alakzatról beszélünk (2), ahol nem használhatunk LRS-t (0), a 03-as alakzat pedig egy poligon.

Fontos tudnunk, hogy az itt megadott dimenzióadat meghatározza például azt is, hogy hány adat ad meg egy pontot vagy egy egyenest, illetve hogy egy rétegen belül nem keverhetjük a különböző dimenziószámú elemeket.

*SDO_SRID*: a koordinátarendszert adja meg. Ha értéke NULL, akkor nem definiálunk koordinátarendszert. Ha nem NULL, akkor egy létező értéknek kell lennie az *SDO_COORD_REF_SYS* táblából, továbbá ennek az *SRID* értéknek szerepelnie kell a metadatok között is. Egy oszlopban lévő alakzatoknak azonos *SRI*D értéket kell adnunk.

*SDO_POINT*: összetett adatszerkezet, az alábbi típusdefinícióval:

```
CREATE TYPE sdo_point_type AS OBJECT (
   X NUMBER,
   Y NUMBER,
   Z NUMBER);
```

Egy darab háromdimenziós pontot tudunk vele definiálni. Ha az utolsó két elem az *SDO_GEOMETRY*-ben NULL, akkor ez a mező nem NULL, X és Y értéke egy pont koordinátáinak felel meg. Ha az említett mezők értéke nem NULL, akkor a Spatial ezt a mezőt figyelmen kívül hagyja. Pont típusú alakzatot tárolhatnánk a másik két mezőben is, azonban csak pont típusú alakzat esetén ez javallott.

*SDO_ELEM_INFO*: összetett adatszerkezet az alábbi típussal:

```
CREATE TYPE sdo_elem_info_array AS VARRAY (1048576) of NUMBER;
```

Tehát egy változó hosszúságú tömbről van szó, melyben megtudhatjuk, hogy az *SDO_ORDINATES* tömbben tárolt adatokat hogyan kell értelmezni. Minden egyes, az objektumunkban tárolt alakzathoz egy számhármas tartozik, amely az alábbi formátumú:

```
(SDO_STARTING_OFFSET, SDO_ETYPE, SDO_INTERPRETATION)
```

Az első mutatja meg, hogy az alakzathoz tartozó koordinátaértékek hol kezdődnek az *SDO_ORDINATES* tömbben, a számozás az 1-gyel indul. *SDO_ETYPE* az alakzat típusát adja meg, vagyis hogy egyszerű vagy összetett-e, illetve megadható az is egy poligon esetén például, hogy az egyes pontokat az óramutató járásával megegyezően vagy azzal ellentétesen adunk-e meg. Az utolsó szám két dolgot jelenthet. Összetett alakzat esetén azt mutatja meg, hogy hány darabból épül fel az alakzat, míg egyszerű esetben azt tudhatjuk meg, hogy miként értelmezzük a megadott pontkoordinátákat (vagyis hogy például a megadott ponthalmazt, mint törtvonalat, vagy pedig mint körívekből álló alakzatot kell értelmezni).

*SDO_ORDINATES*: ez a mező tartalmazza az alakzatot felépítő pontok koordinátáit. Attól függően, hogy hány dimenziós alakzatot írunk le, változik az egy ponthoz tartozó elemek száma, tehát például 3 dimenziós esetben egy ponthoz 3 érték tartozik. Az egy ponthoz tartozó koordinátértékeket rendezetten tároljuk, tehát a koordinátáknak mindig ugyanaz a sorrendje.

Alapértelmezésben a Spatial nem ellenőrzi, hogy a *GTYPE*-ban megadott alakzat típusnak megfelelnek-e az itt megadott koordináták, ezt nekünk kell figyelemmel kísérnünk. Erre a célra létezik egy függvény is, ez a *SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT*.

## Metódusok

Az *SDO_GEOMETRY*-hez definiáltak alapértelmezett metódusokat, melyekkel az adatszerkezet különböző részeit lehet lekérdezni. Létezik ilyen metódus a dimenziószám, a *GTYPE* típus és az LRS adatainak lekérdezésére, továbbá annak eldöntésére, hogy a megadott alakzatunk érvényes-e. Két további metódus segítségével pedig (*GET_WKB*, *GET_WKT*) bináris illetve szöveges formátumba alakíthatjuk át az objektumunkat, ezeket az alakokat a konstruktorok tudják felhasználni. Ilyenkor az objektum minden adata átkerül az új formátumba, viszont a koordinátarendszer információi elvesznek.

## Konstruktorok

Konstruktorok segítségével geometriai alakzatot hozhatunk létre az előző részben már említett bináris illetve szöveges formátumból. Mivel ezek a formátumok nem tartalmaznak koordinátarendszerrel kapcsolatos információkat, ezért ezt nekünk kell beállítanunk, ügyelve arra, hogy ez az adat illeszkedjen a már korábban definiált alakzatok koordinátarendszeréhez.

## Metaadatok

A geometriai metaadatok a dimenziók jellemzésére használatosak, minden egyes dimenzióhoz tárolhatjuk a magasabb és alacsonyabb szintek felé fennálló kötöttségeket illetve az adott dimenzió tűrését is. Ezek az adatok egy külön táblában tárolódnak, melynek tulajdonosa *MDSYS*. Minden Spatial felhasználó két nézetet láthat ebből a táblából, ezek: *USER_SDO_GEOM_METADATA* és *ALL_SDO_GEOM_METADATA*. Az előbbi tartalmazza azokat a metaadatokat, melyek a felhasználó saját tábláihoz kapcsolódnak. Ebben a nézetben lehetőségünk van módosítani és beszúrni is. A második nézet minden táblához tárolja a metaadatokat, ez a nézet lekérdezésekhez használható.

Egy metaadat az alábbi definícióval rendelkezik:

```
( TABLE_NAME    VARCHAR2(32),
  COLUMN_NAME   VARCHAR2(32),
  DIMINFO       SDO_DIM_ARRAY,
  SRID          NUMBER
);
```

A tábla és az oszlop neve adja meg, hogy hol található a térbeli / geometriai információ. A *DIMINFO* egy változó hosszúságú (max. 4) tömb, ahol minden dimenzióhoz egy adatszerkezetet tárolunk, egy ilyen adatszerkezet a dimenzióra jellemző értékeket tárolja. *SRID* mezőben pedig a koordinátarendszer típusát találhatjuk, ha értéke NULL, akkor nincs meghatározva koordinátarendszer. A mindenki által látható metaadatok esetén a definíció egy *OWNER* mezővel egészül ki, itt a táblát birtokló felhasználó található.

A Spatial által használt indexelési eljárásnak is szüksége van metaadatokra, ezeket szintén egy közös táblában tárolja az Oracle. A felhasználók itt is két nézetben láthatják az adatokat, az egyikben a saját tábláikhoz tartozó indexeket módosíthatják, míg a másikban minden felhasználó adata látható, de ez csak lekérdezhető.

## Geokódolás

Geokódolásnak nevezzük azt az eljárást, amikor megfeleltetést létesítünk földrajzi / térbeli pontok (például hosszúsági és szélességi koordinátáival adott pontok) és postai címek között. Egy postai cím lehet formázott vagy formázatlan. Formázott esetben a cím egy attribútumhalmaz, melyben olyan elemek találhatók, int utca neve, házszám, ország, település, irányítószám, stb. Formázatlan esetben a cím a sztring, amelynek formátuma nem kötött, lehetnek benne olyan elemek is, melyek a geokódolás számára feleslegesek.

Ilyen alkalmazások során gyakran előfordul, hogy nem tudunk pontos címekkel dolgozni. Ezért vezettek be illeszkedési módokat, amik azt határozzák meg, hogy a megadott (és nem feltétlenül pontos) címnek mennyire kell illeszkednie a geokódolt címmel. Ez meglehetősen széles határok között változhat, kezdve a teljes illeszkedéstől egészen odáig, hogy már csak azt követeljük meg, hogy a két helyszín azonos megyében legyen. Alapértelmezett esetben az utcanévben engedünk csak meg eltéréseket.

Geokódoláshoz egy új adattípust is definálták, ez az *SDO_GEO_ADDR* típus. Ilyen típusú változóban tudunk formázott címet eltárolni, illetve olyan eljárások, melyek kimenete valamilye címinformáció, szintén ilyen típust használnak. Gyakorlatilag minden lehetséges címattribútumot tartalmaz, ezek közül természetesen nem használjuk mindig az összest. Kiegészítették még olyan mezőkkel, melyek az illeszkedési módot és az esetleges hibaüzenetet tárolják (eljárások kimenete számára). Az utolsó két mező pedig a geokódolás eredménye, vagyis egy hosszúsági és szélességi koordináta érték.

Geokódolás segítségével többféle konverziót is végezhetünk adatainkon, léteznek függvények melyekkel formázott vagy formázatlan címeket tudunk kódolni, van lehetőség inverz műveletre, vagyis amikor földrajzi adatokból kapunk címinformációt, továbbá a kimeneti típus is változtatható, ha szükséges, szabványos geometriai típus is lehet egy kódolás eredménye.

A geokódoláshoz szükséges adatokat többféle táblából nyerhetjük ki. Létezik két olyan tála, amelyet az Oracle biztosít számunkra, ezek címformátumokat tartalmaznak. Vannak továbbá olyan táblák is, melyek országspecifikusak, itt olyan adatok tárolódhatnak, mint irányítószámok, utcanevek, helységnevek, kereszteződések, stb.

## Koordinátarendszerek

Koordinátarendszernek vagy térbeli viszonyítási rendszernek nevezzünk egy olyan szabályrendszert, mely koordináták halmazát rendeli egy térbeli ponthoz. Többféle ilyen rendszer használatos, a legismertebbek:

- Descartes koordinátarendszer: egy meghatározott pontból (az origóból) kiinduló két (vagy három) merőleges egyenes által megadott viszonyítási alap
- geográfiai koordináták: tulajdonképpen a szélességi és hosszúsági adatok, melyek nagyban hasonlítanak a matematikai polárkoordinátákra
- projektív koordináták: síkbeli Descartes-féle koordinátarendszer, amit úgy kaphatunk, hogy egy földfelszíni pontból a matematikai vetítés műveletével leképezzük a koordinátarendszert egy síkra
- helyi koordináták: olyan Descartes-féle koordinátarendszer, amely független a Földtől

Ezek között létezik egy egyértelmű megfeleltetés, tehát egyik koordinátarendszerből egyértelműen át tudunk térni egy másikra.

A Spatial-ben lehetőségünk van tetszőleges koordnátarendszerben értelmezni az adatainkat, a különböző megvalósítások közötti átjárás teljes körűen biztosított.

A koordinátarendszer választás tehát teljesen ránk van bízva, azonban van néhány olyan tényező, mely igen fontos lehet a választás során. Az egyik ilyen a kezelt terület mérete, mivel földrajzi koordinátáink nagy területen is pontos eredményt adnak, ha viszont Descartes koordinátákat használunk a Földön nagy távolságok mérésére, akkor megehetősen nagy hibát is okozhatunk. Másrészről viszont Descartes koordinátákkal sokkal könnyebb, ezáltal gyorsabb is számolni, így ha fontos a gyorsaság és némi pontatlanság megengedett, akkor célszerűbb Descartes koordinátákat használni földrajzi hosszúsági és szélességi adatok helyett.

## Újdonságok

Spatial és Locator újdonságok Oracle 10*g*-ben

Locator:

- European Petroleum Surve Group (EPSG) adatmodelljének támogatása
- explicit koordinátatranszformációk
- egyszerűsített operátorok
- párhuzamos lekérdezések

Spatial:

- topológiai és hálózati adatmodell fejlesztése
- geokódoló motor
- GeoRaser tömörítési algoritmus
- útvonaltervező bővítése Nyugat-Európával
- térinformatikai analitikus függvények

Spatial és Locator újdonságok Oracle 11*g*-ben

Locator:

- puffer függvények
- SQL/MM térbeli függvények és típusok támogatása

Spatial:

- új 3D Java adatmodell
- GeoSpatial Web Services
- GeoRaster fejlesztése (több fájlformátum, adat- és metaadattípus, megbízhatóság)
- hálózati adatmodell fejlesztése
- útvonaltervező motor fejlesztése
- SQL/MM térbeli függvények és típusok támogatása

# Business Intelligence – Üzleti intelligencia áttekintés

Az Oracle BI egy átfogó portfolió, mely alaptechnológiákat és hozzájuk kapcsolódó alkalmazásokat fog össze. Ezekkel az elemekkel a teljes vállalatirányítást átfogja, vagyis olyan képességekkel ruházza fel az adatbázis-kezelőt, melyekkel a teljes vállalatot érintő összes adatot és összefüggést egyszerre átláthatjuk, megkönnyítve ezzel a megfelelő intézkedések meghozatalát és vállalatirányítási folyamatok indítását.

Ez az alrendszer az Oracle köztesszoftver termékcsaládjának tagja, így teljes körűen integrálható az Oracle termékeihez, legyen az új telepítésű, vagy akár már használatban lévő rendszer. Az adatbázis-kezelőkhöz, köztesszoftverekhez és alkalmazásokhoz is optimalizált megoldások segítségével egy teljesen egységes BI rendszert hozhatunk létre, így jelentősen egyszerűsíthetjük a menedzsmentet és csökkenthetjük a költségeinket. Az integrálás megoldható nem Oracle termékek esetén is a szabványosított kapcsolódó felületeken keresztül, így a felhasználás köre még szélesebb lehet.

## Mi az az üzleti intelligencia?

Az üzleti intelligencia egy folyamat melynek során információt szerzünk vállaltunk működéséről, állapotáról, eredményességéről, stb. a rendelkezésünkre álló adatok alapján. Egy mai vállalat rengeteg információt gyűjt és tárol a különböző tranzakciós rendszerein keresztül. Ezek az adatok kiválóan alkalmasak az üzleti intelligencia analíziseihez.

## Business Intelligence felépítése

A vállalatok számára szükséges széleskörű rálátás biztosításhoz sok, eddig nem használt képességet használ a BI. Az információk minden felhasználó számára fontosak, így igen sok formában kell tudnunk az összegyűjtött adatokat továbbadni, ilyenek például a különböző riportok, riasztási listák, dashboardok. Lényeges szempont, hogy az információt a kellő időben tudjuk prezentálni, ne teljen el feleslegesen sok idő, mire az összegyűjtött adatokból használható információ válik. Szintén egy fontos elem, hogy a már meglévő üzleti folyamatokba, workflow-kba észrevétlenül integrálódjanak az alkalmazás elemei, ne kelljen ezeket lényegesen módosítani. A BI architektúra továbbá alapot kell adjon ahhoz is, hogy skálázható maradjon a rendszerünk a nagyon gyorsan növekvő adatmennyiség és a folyamatosan bővülő felhasználói igények mellett is.

Mindezek kielégítésére a BI alrendszer szinte minden rétegében bővíti az Oracle adatbázis-kezelő eredeti szolgáltatáscsomagját. A bővítmények és új fejlesztések 6 nagy kategóriába sorolhatók be:

- Prezentációs eszközök: olyan új elemek, mint például Interactive Dashboard, Answers, Reporting and Publishing, Delivery, Disconnected Analytics
- BI szerver: teljesítmény, skálázhatóság és rendelkezésre állás javítása
- Telepítési és adminisztrációs költségek: mind az időszükséglet, mind az anyagi ráfordítás mértéke csökkent
- Integrálhatóság: alacsonyabb költség és egyszerűbb megoldások
- Adatforrások integrálása: gyorsabb kapcsolódás a vállalati adatforrásokhoz
- Biztonság, azonosítás menedzsment, audit

## Prezentációs eszközök

### Interactive Dashboard

A dashboard egy olyan megjelenítési forma, amelyet minden felhasználó teljesen a saját igényeire szabhat, továbbá amin keresztül minden számára releváns információ elérhető. A nagyfokú rugalmasságnak köszönhetően a dashboardban a felhasználó létrehozhat új típusokat, változókat, ezek értékeit nyomon követheti, majd ezen számított értékekből könnyen érthető kimutatásokat, táblázatokat készíthet.

Látványos újítás például, hogy egy grafikon oszlopait az általuk reprezentált érték alapján színezhetjük, egy így készített grafikon könnyen átlátható. A különböző elemzések során a felhasználó nem csak a végeredményt látja, hanem a folyamatot is, ahogy az kialakul. Eközben interaktívan lehetősége van módosításokra is, meggyorsítva és hatékonyabbá téve ezzel az analízis folyamatát.

Az elemzések készítése során lehetőség van pillanatképeket készíteni az aktuális állapotról, ezeket nevezik „Briefing Book”-nak. A pillanatképekkel könnyen dokumentálható az elemzés folyamata.

### Answers

Az Oracle Answers egy könnyen kezelhető webes lekérdező-felület. A felhasználók számára egy logikai adatstruktúrát mutat, melyen keresztül az adminisztrátorok által kontrolált módon tud a felhasználó az adatokhoz hozzáférni. Ezzel megelőzhetőek a különösen hosszú, idő- és teljesítményigényes lekérdezések okozta problémák, viszont mindenki számára lehetőséget adunk az adatokhoz való hozzáféréshez.

A lekérdezések összeállításához egy előre definiált elemekből építkező, drag-and-drop rendszerű felület is rendelkezésre áll.

### Reporting and Publishing

Ez a szolgáltatás lehetővé teszi, hogy az analízisek során keletkezett eredményeinket a lehető legkülönbözőbb formában mutathassuk be. Többféle dokumentáció-formátumot támogat, és ezeken belül is változatos sablonokat kínál a felhasználóknak. Támogatja többek között az Adobe Acrobat és a Microsoft Office dokumentum-formátumait. Sablon segítségével készíthetünk ezen belül pénzügyi jelentés, vállalati adatlapokat, riportokat, címkéket, megrendelőket. Az elkészült dokumentumok egyszerűen nyomtathatók, küldhetőek e-mailben vagy faxon, vagy akár közzétehetők web-en is.

### Delivers

Az Oracle Delivers egy komplex felügyeleti és riasztási rendszer. Folyamatosan monitorozza az összes a vállalathoz tartozó adatforrást. Ennek során az adminisztrátorok által beállított előre definiált nem kívánt eseményeket felismerve automatikusan jelentést generál, majd értesíti a megfelelő személyeket a bekövetkezett eseményről. Az értesítés történhet tetszőleges eszközön (email, mobil eszköz, vagy csak egyszerűen egy monitoron keresztül).

## BI szerver

Egy átlagos vállalat alulról felfelé építkezve állítja össze üzleti alkalmazásait. Ez azt jelenti, hogy először létrehozzák az adattárházakat, majd ezekre építve fokozatosan jutnak el a különböző analizáló eszközökig, végül pedig az analízis eredményéhez. Az Oracle BI ezzel merőben ellentétes elveket követ. Először felméri, hogy a vállalat számára milyen adatok, információk a legfontosabbak, majd ezen felmérés eredményeként határozza, meg, hogy hogyan történjen az adatok tárolása, az aggregáció, a különböző adatforrások összerendelése. Ezáltal jelentős teljesítmény-növekedést kapunk, különösen olyan esetekben, ahol nagyon nagy mennyiségű adatot kell tárolni, és elemezni, és egyszerre sok adatforrásból kapjuk az információt. Ezzel a technológiával megkönnyíthetjük nem Oracle alapú alkalmazások integrációját is, hiszen ezeket már a tervezés korai fázisában figyelembe tudjuk venni.

További teljesítmény-növekedést érhetünk el az új cache-elési infrastruktúrával. A fürtözött rendszerben lévő szerverek nem csak saját maguknak tárolják az információt, hanem körözvények segítségével minden másik szerverhez is eljuttatják információikat, ezáltal sokkal gyorsabbá téve az adatok elérését.

Javult a konzisztencia-ellenőrzés is, az adminisztrátoroknak lehetősége van egyenként ellenőrizni az objektumokat, és gyorsan visszaküldeni azokat a felhasználónak javításra.

A BI szervereknél a legjobb teljesítményt úgy lehet elérni, ha ismerjük az adott adathierarchia rétegeinek számát. Egy új fejlesztés eredményeként ez automatikusan beállítódik, tovább optimalizálva a lekérdezéseket.

## Telepítési és adminisztrációs költségek

Több újítás segítségével sikerült csökkenteni a különböző költségeken:

- többszörözött prezentációs szerverek segítségével kiegyensúlyozottabbá vált a forgalom.
- manapság igen elterjedt mobil eszközökhöz igazodva létezik a kapcsolat nélküli analízis fogalma, így egy mobil eszköz akkor is végezhet elemzéseket, amikor nincs kapcsolódva a vállalati rendszerhez. Ehhez meg kellett oldani a szinkronizációs problémákat a vállalati hálózat többi elemével, méghozzá úgy, hogy a szinkronizáció láthatatlan legyen.

## Integrálhatóság

A BI nagyfokú integrálhatósága érdekében a fejlesztők sokféle interfésszel látták el, mellyel könnyedén tud kapcsolódni más rendszerekhez. Talán a legfontosabb, hogy a vállalatirányítási rendszerek által használt SOA (Service-Oriented Architechture) és BPEL (Business Process Execution Language) paradigmákat felismeri és támogatja. Emellett megvalósításra került a BI-ben az Oracle több terméke által használt VPD (Virtual Private Database), Internet Directory, SSO (SingleSign-On) és EBS (e-Business Suite) Security Modell is.

## Adatforrások integrálása

Az Oracle BI nyílt szabványokra épül, így elvileg tetszőleges programmal csatlakozhatunk hozzá. Ez a szoftver lehet adatforrás, biztonsági modell, alkalmazás vagy tetszőleges végfelhasználói eszköz. Támogatott eszközök például az SAP vállalatirányítási szoftverei, OLAP motorok, Microsoft Analysis Services programcsomag, továbbá olyan adatbázis-kezelők, mint az IBM DB2, a Teradata vagy a Microsoft SQL Server.

## Biztonság, azonosítás menedzsment, audit

Dashboard Proxy: egy olyan azonosítási mód, melynek során a proxyt használó felhasználó egy másik felhasználó helyébe lép, vagyis pontosan ugyanazt látja, mint ő. Ez akkor lehet például hasznos, amikor hibaelhárítás során szeretnénk látni a hibát az eredeti környezetében, de abban az esetben is előnyös, ha csak meg szeretnénk osztani olyan információkat, amihez csak nekünk van hozzáférésünk.

Az adminisztrátoroknak lehetősége van bármely két, a BI hálózatában lévő gép közötti adatforgalmat SSL-en keresztül titkosítani, biztonságosabbá téve ezzel az esetleges érzékeny adatok továbbítását, kezelését.

# Spatial a gyakorlatban

A korábban összefoglalt elméleti ismeretanyagot mindenképpen szerettem volna kipróbálni a gyakorlatban is. Sajnos az időm rövidsége és beállítási problémák miatt nem tudtam megoldani, hogy egy előre elkészített adatbázist betöltsek, majd ezen végezzek vizsgálatokat. Ezt azonban a későbbiek során mindenképpen pótolni szeretném.

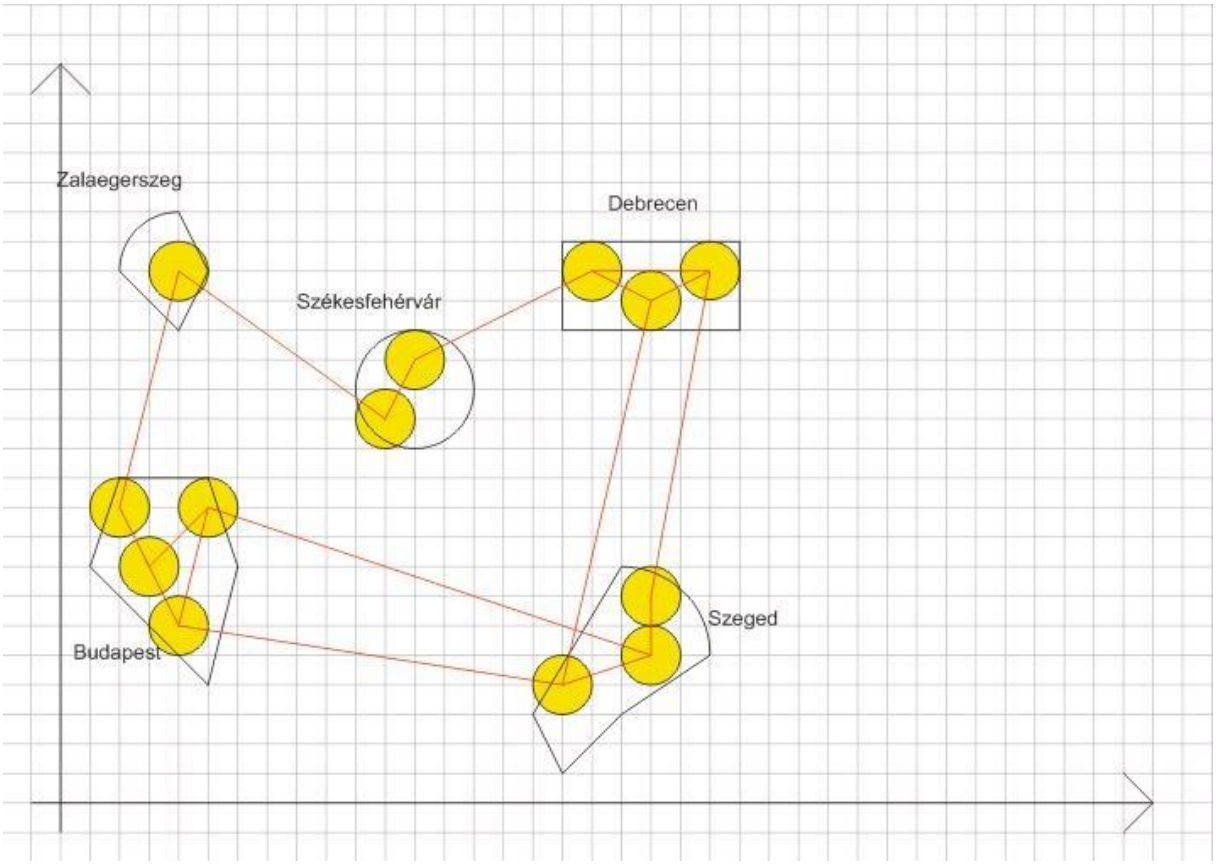
Így azonban egy általam kreált kis mintapélda volt az alapja a gyakorlati vizsgálataimnak.

## Mintapélda

Egy képzeletbeli magyar távközlési szolgáltató internetelérést biztosít Magyarországon 5 szintén képzeletbeli városban, ezek Budapest, Zalaegerszeg, Székesfehérvár, Debrecen és Szeged. A szolgáltatónak van egy térképe, melyen fel vannak tüntetve a városok, a városokon belüli csomópontok, és a csomópontok közötti fizikai összeköttetések. Azt tudjuk még, hogy egy csomópont 1 km-es körzetében tud a szolgáltató szolgáltatni.

A szolgáltató ezeket az adatokat egy adatbázisban tárolja, és különböző összefüggések érdeklik, például milyen messze vannak egymástól a városok, vagy mekkora egy város lefedettsége.

A szolgáltató térképe így néz ki:



A megoldáshoz egy sql szkriptet írtam mely alant olvasható (a sorok feltöltésénél nem írtam ki mindent, hogy egy picit rövidebb legyen):

```
-- Halozat.sql

-- Táblák eldobása

DROP TABLE cities;

DROP TABLE nodes_cover;

DROP TABLE paths;

DROP TABLE nodes;

DELETE FROM user_sdo_geom_metadata;

-- Táblák létrehozása

CREATE TABLE cities (
  id NUMBER PRIMARY KEY,
  name VARCHAR2(16),
  shape SDO_GEOMETRY);

CREATE TABLE nodes (
  id NUMBER PRIMARY KEY,
  name VARCHAR2(5),
  shape SDO_GEOMETRY);

CREATE TABLE nodes_cover (
  id NUMBER PRIMARY KEY,
  shape SDO_GEOMETRY);

CREATE TABLE paths (
  id NUMBER PRIMARY KEY,
  name VARCHAR2(12),
  startpoint NUMBER,
  endpoint NUMBER,
  shape SDO_GEOMETRY);

ALTER TABLE paths ADD (
  CONSTRAINT "start" FOREIGN KEY(startpoint)
      REFERENCES nodes(id),
  CONSTRAINT "end" FOREIGN KEY(endpoint)
      REFERENCES nodes(id));

ALTER TABLE nodes_cover ADD (
  CONSTRAINT "id" FOREIGN KEY(id)
        REFERENCES nodes(id));

-- Sorok létrehozása

-- Városok

INSERT into cities VALUES(1, 'Budapest', SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1003,1), SDO_ORDINATE_ARRAY(5,4, 6,8, 5,11, 2,11,
1,8, 5,4)));

INSERT into cities VALUES(2, 'Zalaegerszeg', SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1005,2, 1,2,1, 7,2,2), SDO_ORDINATE_ARRAY(2,18, 4,16,
5,18, 4,20, 2.586,19.414, 2,18)));
```

```
INSERT into cities VALUES(3, 'Szekesfehervar', SDO_GEOMETRY(2003, NULL,
NULL, SDO_ELEM_INFO_ARRAY(1,1003,4), SDO_ORDINATE_ARRAY(9,14, 11,12,
13,14)));

INSERT into cities VALUES(4, 'Debrecen', SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1003,3), SDO_ORDINATE_ARRAY(17,16, 23,19)));

INSERT into cities VALUES(5, 'Szeged', SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1005,2, 1,2,1, 9,2,2), SDO_ORDINATE_ARRAY(19,8,
16,3, 17,1, 19,3, 22,5, 21.121,7.121, 19,8)));

COMMIT;

-- Csomópontok

INSERT INTO nodes VALUES(1, 'BP1', SDO_GEOMETRY(2001, NULL,
SDO_POINT_TYPE(2,10,NULL), NULL, NULL));
INSERT INTO nodes VALUES(2, 'BP2', SDO_GEOMETRY(2001, NULL,
SDO_POINT_TYPE(3,8,NULL), NULL, NULL));

...

INSERT INTO nodes VALUES(13, 'SZEG3', SDO_GEOMETRY(2001, NULL,
SDO_POINT_TYPE(17,4,NULL), NULL, NULL));

COMMIT;

-- Utak

INSERT INTO paths VALUES(1, 'BP1-BP2', 1, 2, SDO_GEOMETRY(2002, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,2,1), SDO_ORDINATE_ARRAY(2,10, 3,8)));
INSERT INTO paths VALUES(2, 'BP2-BP3', 2, 3, SDO_GEOMETRY(2002, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,2,1), SDO_ORDINATE_ARRAY(3,8, 5,10)));

...

INSERT INTO paths VALUES(17, 'SZEG3-BP4', 13, 4, SDO_GEOMETRY(2002, NULL,
NULL, SDO_ELEM_INFO_ARRAY(1,2,1), SDO_ORDINATE_ARRAY(17,4, 4,6)));

COMMIT;

-- Lefedettség feltöltése

INSERT INTO nodes_cover VALUES(1, SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1003,4), SDO_ORDINATE_ARRAY(1,10, 2,9, 3,10)));
INSERT INTO nodes_cover VALUES(2, SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1003,4), SDO_ORDINATE_ARRAY(2,8, 3,7, 4,8)));

...

INSERT INTO nodes_cover VALUES(13, SDO_GEOMETRY(2003, NULL, NULL,
SDO_ELEM_INFO_ARRAY(1,1003,4), SDO_ORDINATE_ARRAY(16,4, 17,3, 18,4)));

COMMIT;

-- Metaadatok betöltése
INSERT INTO user_sdo_geom_metadata
    (TABLE_NAME,
     COLUMN_NAME,
     DIMINFO,
     SRID)
```

```
  VALUES (
  'cities',
  'shape',
  SDO_DIM_ARRAY(
    SDO_DIM_ELEMENT('X', 0, 30, 0.005),
    SDO_DIM_ELEMENT('Y', 0, 30, 0.005)
     ),
  NULL);

...

INSERT INTO user_sdo_geom_metadata
    (TABLE_NAME,
     COLUMN_NAME,
     DIMINFO,
     SRID)
  VALUES (
  'nodes_cover',
  'shape',
  SDO_DIM_ARRAY(
    SDO_DIM_ELEMENT('X', 0, 30, 0.005),
    SDO_DIM_ELEMENT('Y', 0, 30, 0.005)
     ),
  NULL);

COMMIT;
-- Indexek létrehozása

CREATE INDEX cities_index
   ON cities(shape)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;

CREATE INDEX nodes_index
   ON nodes(shape)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;

CREATE INDEX paths_index
   ON paths(shape)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;

CREATE INDEX nodes_cover_index
   ON nodes_cover(shape)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;

COMMIT;

-- Lekérdezések

-- Városok területei

SELECT name as city, SDO_GEOM.SDO_AREA(shape, 0.005) as area, 'km^2' as
measure
FROM cities;

-- Melyik csomópont melyik városban van?

SELECT node, city
FROM (SELECT n.name as node, c.name as city, SDO_GEOM.RELATE(c.shape,
'anyinteract', n.shape, 0.005) as inside
FROM cities c, nodes n)
WHERE inside = 'TRUE';
```

```
-- Két város közötti fizikai kapcsolatok?

SELECT c1.name as city1, c2.name as city2, p.name as path_name
FROM cities c1, cities c2, nodes n1, nodes n2, paths p
WHERE NOT(c1.id=c2.id) AND
    SDO_GEOM.RELATE(c1.shape, 'contains', n1.shape, 0.005) = 'CONTAINS' AND
    SDO_GEOM.RELATE(c2.shape, 'contains', n2.shape, 0.005) = 'CONTAINS' AND
    p.startpoint = n1.id AND
    p.endpoint = n2.id;

-- Fizikai kapcsolatok hossza?

SELECT p.name as path, SDO_GEOM.SDO_LENGTH(p.shape, m.diminfo) as length,
'km' as km
FROM paths p , user_sdo_geom_metadata m
WHERE m.table_name = 'PATHS' AND m.column_name = 'SHAPE';

-- Városok közötti távolságok?

SELECT c1.name as city1, c2.name as city2, SDO_GEOM.SDO_DISTANCE(c1.shape,
c2.shape, 0.005) as distance, 'km' as km
FROM cities c1, cities c2
WHERE c1.id < c2.id;


-- Debrecen lefedettsége?

SELECT city, SDO_GEOM.SDO_AREA(SDO_GEOM.SDO_DIFFERENCE(shape, cover,
0.005), 0.005) / SDO_GEOM.SDO_AREA(shape, 0.005) * 100 as coverage, '%'
as meas
FROM( SELECT c.name as city,
      SDO_GEOM.SDO_UNION(SDO_GEOM.SDO_UNION(nc1.shape, nc2.shape, 0.005),
      nc3.shape, 0.005) as cover, c.shape as shape
      FROM cities c, nodes n1, nodes n2, nodes n3, nodes_cover nc1,
      nodes_cover nc2, nodes_cover nc3, user_sdo_geom_metadata m
      WHERE n1.id = nc1.id AND
            n2.id = nc2.id AND
            n3.id = nc3.id AND
            n1.id < n2.id AND
            n1.id < n3.id AND
            n2.id < n3.id AND
            SDO_GEOM.RELATE(c.shape, 'contains', n1.shape, 0.005) =
              'CONTAINS' AND
            SDO_GEOM.RELATE(c.shape, 'contains', n2.shape, 0.005) =
              'CONTAINS' AND
            SDO_GEOM.RELATE(c.shape, 'contains', n3.shape, 0.005) =
              'CONTAINS' AND
            c.name = 'Debrecen' AND
            m.table_name = 'NODES_COVER' AND
            m.column_name = 'SHAPE');
```

A kimenet pedig az alábbiak szerint alakult (a kommentek utólagosan kerültek bele a jobb átláthatóság kedvéért):

```
-- Városok területe:

CITY                    AREA MEAS
--------------- ---------- ----
Budapest                  22 km^2
Zalaegerszeg     7,14096678 km^2
Szekesfehervar   12,5663706 km^2
Debrecen                  18 km^2
Szeged           20,5671752 km^2

-- Melyik csomópont melyik városban van?

NODE  CITY
----- ---------------
BP1   Budapest
BP2   Budapest
BP3   Budapest
BP4   Budapest
ZEG1  Zalaegerszeg
SZFV1 Szekesfehervar
SZFV2 Szekesfehervar
DEB1  Debrecen
DEB2  Debrecen
DEB3  Debrecen
SZEG1 Szeged
SZEG2 Szeged
SZEG3 Szeged

-- Városok között milyen összeköttetések vannak?

CITY1            CITY2            PATH_NAME
---------------- ---------------- ------------
Budapest         Zalaegerszeg     BP1-ZEG1
Zalaegerszeg     Szekesfehervar   ZEG1-SZFV1
Szekesfehervar   Debrecen         SZFV2-DEB1
Debrecen         Szeged           DEB2-SZEG1
Debrecen         Szeged           DEB3-SZEG3
Szeged           Budapest         SZEG2-BP3
Szeged           Budapest         SZEG3-BP4

-- Összeköttetések hossza?

PATH             LENGTH KM
------------ ---------- --
BP1-BP2      2,23606798 km
BP2-BP3      2,82842712 km
BP2-BP4      2,23606798 km
BP3-BP4      4,12310563 km
BP1-ZEG1     8,24621125 km
ZEG1-SZFV1   7,81024968 km
SZFV1-SZFV2  2,23606798 km
SZFV2-DEB1   7,61577311 km
DEB1-DEB2             4 km
DEB1-DEB3    2,23606798 km
DEB2-DEB3    2,23606798 km
DEB2-SZEG1   11,1803399 km
DEB3-SZEG3   13,3416641 km
SZEG1-SZEG2           2 km
```

```
SZEG2-SZEG3  3,16227766 km
SZEG2-BP3    15,8113883 km
SZEG3-BP4    13,1529464 km

-- Városok egymástól mért legkisebb távolsága?

CITY1           CITY2              DISTANCE KM
--------------- --------------- ---------- --
Budapest        Zalaegerszeg              5 km
Budapest        Szekesfehervar   4,70820393 km
Budapest        Debrecen                 13 km
Budapest        Szeged           10,9141031 km
Zalaegerszeg    Szekesfehervar   5,15541753 km
Zalaegerszeg    Debrecen                 12 km
Zalaegerszeg    Szeged           16,9783599 km
Szekesfehervar  Debrecen         4,32455532 km
Szekesfehervar  Szeged                    8 km
Debrecen        Szeged                    8 km

-- Debrecen lefedettsége?

CITY              COVERAGE M
--------------- ---------- -
Debrecen        47,6401224 %
```

## Tanulságok

Még egy ilyen egyszerű kis példában is, mint a fent vázolt, sok tapasztalatot lehet szerezni.

Jelentősen megkönnyíti például a lekérdezések összeállítását a Spatial függvények egymásba ágyazhatósága, mivel majdnem minden ilyen függvény visszatérése egy *SDO_GEOMETRY* elem. Ezáltal sokkal tömörebben és átláthatóbban lehet leírni egy lekérdezést, ahogy az az utolsó példánál is látható.

A lekérdezések összeállításánál vált érthetővé számomra a metaadatok jelentősége is. Több függvénynek két alakja is van, az egyik esetben elegendő egy toleranciaértéket megadni a számítás elvégzésére, ez látható például az utolsó lekérdezésnél. Azonban lehetőségünk van a függvény paramétereihez egy metaadatsort is hozzárendelni, így különböző koordinátarendszerekben definiált adatok esetén is el tudjuk végezni a műveletet anélkül, hogy nekünk kellene a konverziót elvégezni.

# Forrás


[1] Spatial User's Guide
[2] http://www.oracle.com/technology/pub/articles/lokitz-spatial-geoserver.html
[3] http://otn.oracle.com
[4] New Release in Oracle Business Intelligence Suite Enterprise Edition – Oracle White Paper
[5] Oracle Business Intelligence Technical Overview – Oracle White Paper
[6] ORACLE SPATIAL 11*g* - Advanced Spatial Data Management for the Enterprise – Oracle Data Sheet
[7] ORACLE LOCATOR ÉS ORACLE SPATIAL OPCIÓ - Térinformatika az Oracle Database 10*g* Release 2-ben – Oracle Adatlap
[8] ORACLE LOCATOR - Location Features in Oracle Database 11*g* – Oracle Feature Overview