Prostorové a XML databáze

Marek Rychlý

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů

2. demonstrační cvičení pro PDB
3. října 2014

Obsah

Cíle cvičení

- Úvod do prostorových databází a Oracle Locator/Spatial.
  (prostorová data, post-relační rozšíření Oracle o prostorové databáze)
- Ukázka uložení dat v Oracle Locator/Spatial, indexování.
  (typ `SDO_GEOMETRY`, prostorová data v tabulkách a jejich indexy)
- Ukázka dotazování prostorových dat v Oracle a JDBC.

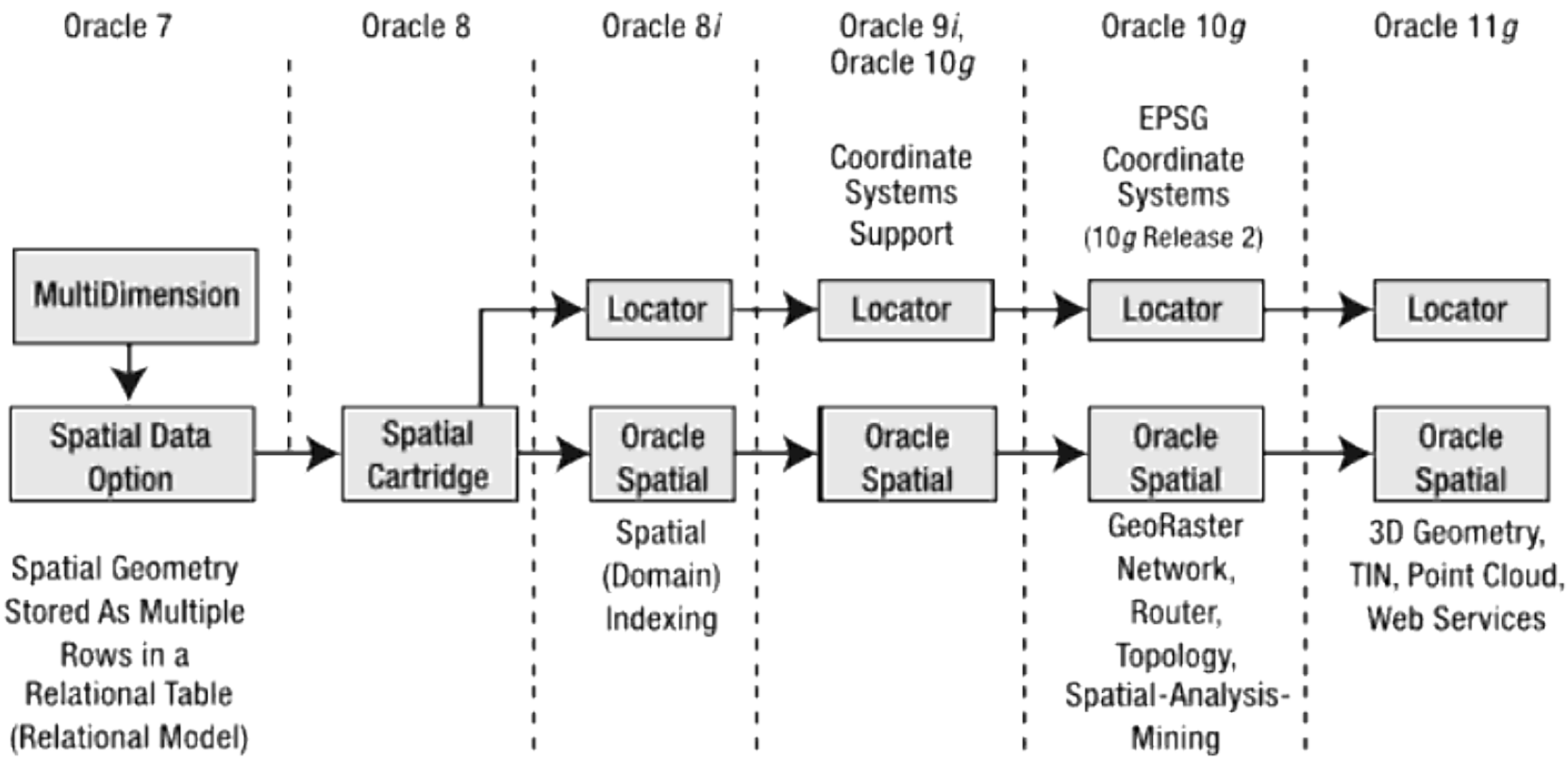
- Úvod do XML databází.
- Ukázka definice XML dat a jejich uložení v Oracle.
- Ukázka dotazování XML dat v Oracle a přes JDBC.

Prostorové databáze

- Prostorová data obvykle zpracovávána specializovanými aplikacemi.
  (GIS, CAD/CAM aplikace, „logistics automation“ systémy, atd.)
- Specializované uzavřené aplikace přinášejí problémy.
  (problémový přístup k prostorovým datům, složitá integrace např. s CRM/ERP)
- Řešení nabízí prostorové databáze.
  (rozšíření klasických relačních databází, např. s daty CRM nebo ERP systémů)
- Prostorové databáze jsou schopny řešit dotazy nad běžnými daty.
  (zaměření kampaně na zákazníky ve spádové oblasti dané prodejny, optimalizace sítě prodejen podle dostupnosti pro zákazníky, plánování tras rozvážky zboží k zákazníkům, atd.)
- Uplatnění v různých oblastech.
  (telekomunikace, státní správa a samospráva, bezpečnostní složky, zeměměřiči, realitní společnosti, média a reklamní společnosti, atd.)

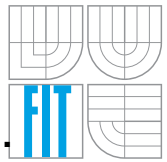
Rozšíření Oracle Locator/Spatial

Oracle podporuje prostorové databáze pomocí rozšíření **Oracle Locator** (základní verze) a **Oracle Spatial** (komerční verze).



Prostorová data v Oracle Locator/Spatial

1. Geometrie – prostorové objekty stejných vlastností ve vrstvách
   - data geometrie ve sloupcích `SDO_GEOMETRY` a `ST_GEOMETRY`,
   - metadata vrstev přes pohledy `*_SDO_GEOM_METADATA`.
     (metadata zahrnují popis dimenzí a souřadnicový systém)
2. GeoRastr – mřížka buněk s hodnotami, pokrývá prostorové objekty
   - data i metadata mřížky ve sloupci `SDO_GEORASTER`,
     (data zahrnují pokrytou geometrii a odkazy na data a metadata buněk,
     metadata mřížky popisují její systém a formát hodnot buněk)
   - data i metadata jednotlivých buněk v tabulce typu `SDO_RASTER`.
     (binární data a metadata se souřadnicemi, přiblížením a oblastí platnosti)
3. Topologie – vrstva popsaná uzly, orient. hranami a stěnami (faces)
   - prvky topologie v tabulkách `*_NODE$`, `*_EDGE$`, `*_FACE$`,
   - tabulky vytvořeny voláním `SDO_TOPO.CREATE_TOPOLOGY`.
4. Sítě – grafy s daty na uzlech, ne/orientované, ne/ohodnocené
   - uzly v tabulce se sloupcem `NODE_ID`, volitelně poloha, cena, atd.
   - hrany v tabulce se sloupcem `LINK_ID`, volitelně poloha, cena, atd.



Datový typ SDO_GEOMETRY

Hodnota typu SDO_GEOMETRY popisuje prostorový objekt.

```
SQL> describe SDO_GEOMETRY
Name                 Null?    Type
-------------------- -------- ------------------------------
SDO_GTYPE                     NUMBER
SDO_SRID                      NUMBER
SDO_POINT                     MDSYS.SDO_POINT_TYPE
SDO_ELEM_INFO                 MDSYS.SDO_ELEM_INFO_ARRAY
SDO_ORDINATES                 MDSYS.SDO_ORDINATE_ARRAY
```

Použití jako typ sloupce tabulky:

```
CREATE TABLE parcela (
   cislo VARCHAR(10) PRIMARY KEY,
   geometrie SDO_GEOMETRY
);
```

Atributy typu SDO_GEOMETRY

- **SDO_GTYPE** – popisuje typ tvaru objektu,
  (point, line, polygon, collection, multipoint, multiline, multipolygon)
- **SDO_SRID** – identifikátor souřadnicového systému pro kóty objektu,
- **SDO_POINT** – souřadnice, pokud je objektem bod,
  (doporučeno pro bod, přestože může být vyjádřen i nasledujícími atributy)
- **SDO_ELEM_INFO** – informace k souřadnicím z SDO_ORDINATES,
  (určení prvních souřadnic; způsob propojení jednotlivých bodů, přímo nebo obloukem; výsledný tvar, jako bod, přímka nebo polygon)
- **SDO_ORDINATES** – souřadnice jednotlivých bodů složitějších objektů.

Pokud je nastavena hodnota atributu SDO_POINT, tak musí být hodnoty atributů SDO_ELEM_INFO a SDO_ORDINATES prázdné (null), a naopak.

Hodnota atributu `SDO_GTYPE` je číslo `DL0T`, kde...

- **D** je celková dimenze objektu ($D \in \{2, 3, 4\}$),
  (počet složek souřadnic objektu vč. případné vzdálenosti v LRS)
- **L** je dimenze pro vzdálenosti v LRS[1] ($L \in \{0, 3, 4\}$),
  (0 znamená žádné LRS, 3 je používá pro 2D a 4 pro 3D objekty)
- **0** je vždy nula (0),
- **T** je druh objektu ($T \in \{0, \ldots, 7\}$):
  - = 0 UNKNOWN_GEOMETRY
  - = 1 POINT
  - = 2 LINESTRING
  - = 3 POLYGON
  - = 4 COLLECTION (points, lines, polygons)
  - = 5 MULTIPOINT
  - = 6 MULTILINESTRING
  - = 7 MULTIPOLYGON

[1] Linear Referencing System (LRS) umožňuje přiřadit bodům jejich vzdálenost od počátku prostorového objektu, jako jejich další rozměr. Používá se pro asociaci dat se vzdáleností, např. uzavírky na dálnici, poruchy na potrubí, atd.

Hodnota atributu `SDO_SRID`...

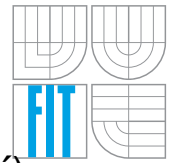
Hodnota atributu `SDO_SRID` je **Spatial Reference System ID**.

- Identifikátor udává souřadnicový systém pro geometrii:
  a) lokální – čistý kartézský souřadnicový systém,
  (pak `SDO_SRID` není nastaveno, tj. null)
  b) geografický – dle modelu zemského tělesa („geodetické datum“).
  (pak `SDO_SRID` je cízí klíč `SRID` z tabulky `MDSYS.CS_SRS`)

- Geografické souřadnicové systémy (kartografická zobrazení):
  i) zeměpisné – mezi 2 body nejkratší spojnice po elipsoidu.
  (ve sloupci `WKTEXT` tabulky `MDSYS.CS_SRS` hodnota „GEOGCS“,
  úhly lon/lat, např. SRID 8307: WGS 84, SRID 4156: S-JTSK)
  ii) rovinné (projekce[2]) – mezi 2 body je nejkratší spojnice přímka,
  (ve sloupci `WKTEXT` tabulky `MDSYS.CS_SRS` hodnota „PROJCS“,
  kartézské jednotky[3], např. SRID 2065: S-JTSK (Ferro) / Krovak)

[2] zeměpisné souřadnice jsou projektovány do roviny kartézské soustavy souřadnic; pro menší oblasti je dostačující (zanedbání změn zakřivení povrchu Země)

[3] např. metry; na rozdíl od úhlových jednotek usnadní výpočet vzdáleností (rychlejší)



Hodnota atributu SDO_POINT...

Hodnota atributu SDO_POINT jsou **souřadnice bodu** (x, y, z).

- U bodů v 2D se třetí souřadnice nenastavuje, tj. null.
- Pro 4D (např. LRS) nutno použít SDO_ELEM_INFO a SDO_ORDINATES.
- Pokud nastaveno SDO_POINT, tak hodnoty atributů SDO_ELEM_INFO a SDO_ORDINATES nesmí být nikdy nastaveny, tj. null.

Příklad[4]:

```
INSERT INTO parcela (cislo, geometrie) VALUES (2,
  SDO_GEOMETRY(2001, 4156,
    SDO_POINT_TYPE(598619, 1157108, NULL),
    NULL, NULL)
);
```

[4] Parcela by neměla být „bod“, ale „polygon“. Naleznete, kde je tato „parcela“?

Atributy SDO_ELEM_INFO a SDO_ORDINATES...

Oba atributy obsahují pole čísel vytvářené procedurami:

- SDO_ELEM_INFO_ARRAY($o_0, t_0, i_0, \ldots, o_n, t_n, i_n$)
- SDO_ORDINATE_ARRAY($x_1, y_1[, z_1, [w_1]], \ldots, x_m, y_m[, z_m, [w_m]]$)

kde pro jednotlivé složky platí, že

- $n = 1$ je jednoduché obekty a $n = k + 1$ pro objekty složené z k částí,
- o je pořadí první souřadnice dané části v poli SDO_ORDINATES,
- t je druh prostorového objektu tvořícího danou část,
- i je interpretace spojení mezi body tvořící objekt dané části,
- x, y, z, w jsou 2D–4D souřadnice m bodů tvořící objekty všech částí.

Jiná realizace bodu z předchozího příkladu:

```
INSERT INTO parcela (cislo, geometrie) VALUES (2,
  SDO_GEOMETRY(2001, 4156, NULL,
    SDO_ELEM_INFO_ARRAY(1,1,1),
    SDO_ORDINATE_ARRAY(598619, 1157108))
);
```

Druhy objektů a interpretace v `SDO_ELEM_INFO`

Druh objektu *t* podobný `SDO_GTYPE`, navíc interpretace dle druhu:

0. UNKNOWN_ELEMENT
1. POINT
   (interpretace: počet bodů, tvořící složený objekt; např. orientace vektoru)
2. LINESTRING
   (interpretace: 1 přímé spoje, 2 obloukové spoje bodů)
3. POLYGON (druh 1003 pro vnější a 2003 pro vnitřní ohraničení)
   (interpretace: 1 přímé spoje, 2 obloukové spoje bodů, 3 obdélník, 4 kružnice)
4. COMPOUND LINESTRING
   (interpretace: počet částí, tvořící složený objekt)
5. COMPOUND POLYGON (1005 pro vnější a 2005 pro vnitřní ohraničení)
   (interpretace: počet částí, tvořící složený objekt)

Druhy 4 a 5 jsou uvedeny v první trojici v `SDO_ELEM_INFO`, další trojice již obsahují popis složek, tj čáry (druh 2) nebo polygony (druh 3).

Vnější a vnitřní ohraničení polygonu a pořadí bodů

Dva druhy ohraničení polygonů:

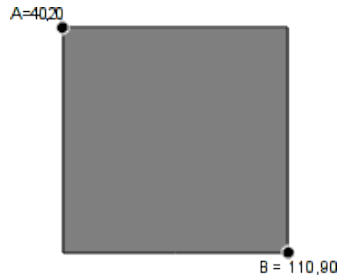
1. $t \in \{1003, 1005\} \Rightarrow$ body polygonu ohraničují jeho obsah z vnějšku
   - polygon je uzavřený,
     (prakticky u všech jednoduchých polygonů)
   - body hranice se zadávají v pořadí proti směru hodinových ručiček.
2. $t \in \{2003, 2005\} \Rightarrow$ body polygonu ohraničují jeho obsah z vnitřku
   - polygon je otevřený,
     (umožňuje polygonem udělat „díru“ do okolí, např. uzavřeného polygonu)
   - body hranice se zadávají v pořadí po směru hodinových ručiček,
   - zadává se až nakonec, po zadání uzavřeného polygonu.

Body obdélníku/kruhu mohou být zadány v libovolném pořadí, ale v dostatečném množství (2 body obdélníku a 3 body kruhu).

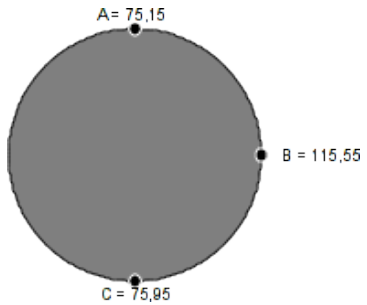
Nesprávné zadání prostorových objektů může vést k jejich nevalidnosti, což způsobí nepoužitelnost v operátorech a funkcích.
(detekce pomocí fce. `ST_IsValid`, bude vysvětleno dále)

Příklady složitějších prostorových objektů I

- **INSERT INTO** parcela (cislo, geometrie) **VALUES** (11,
  SDO_GEOMETRY(2003, **NULL**, **NULL**,
    SDO_ELEM_INFO_ARRAY(1,1003,3),
    SDO_ORDINATE_ARRAY(40,20, 110,90)));

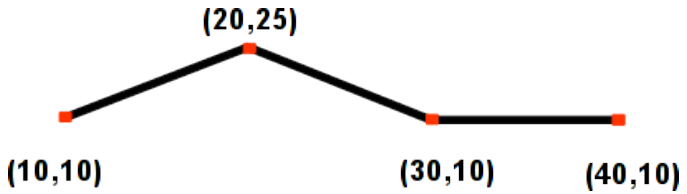


- **INSERT INTO** parcela (cislo, geometrie) **VALUES** (12,
  SDO_GEOMETRY(2003, **NULL**, **NULL**,
    SDO_ELEM_INFO_ARRAY(1,1003,4),
    SDO_ORDINATE_ARRAY(75,15, 75,95, 115,55)));

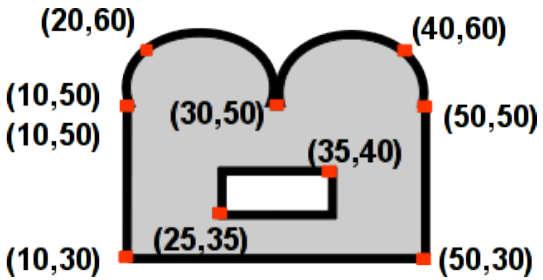


Příklady složitějších prostorových objektů II

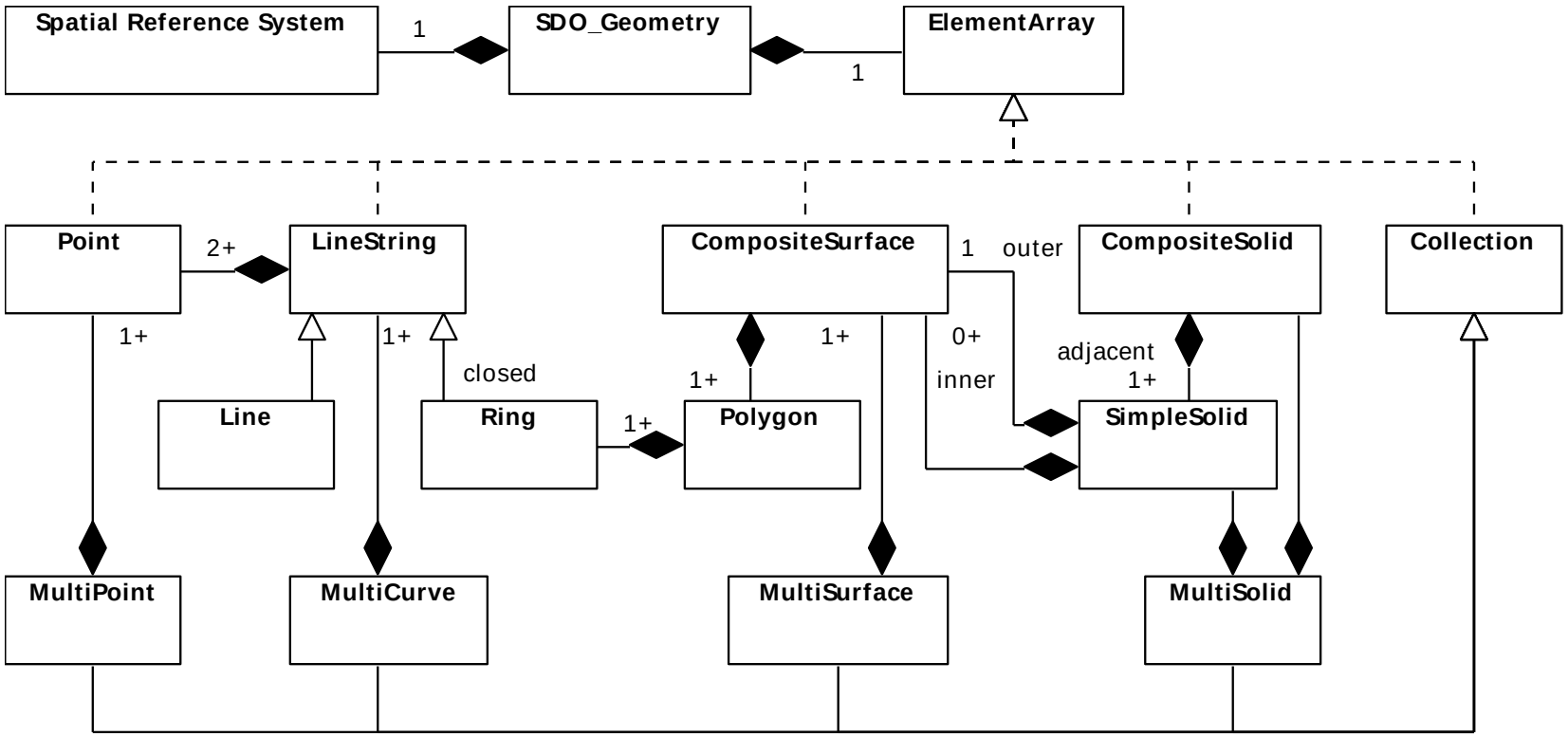
- **INSERT INTO** parcela (cislo, geometrie) **VALUES** (13,
  SDO_GEOMETRY(2002, **NULL**, **NULL**,
    SDO_ELEM_INFO_ARRAY(1,2,1),
    SDO_ORDINATE_ARRAY(10,10, 20,25, 30,10, 40,10)));



- **INSERT INTO** parcela (cislo, geometrie) **VALUES** (14,
  SDO_GEOMETRY(2003, **NULL**, **NULL**,
    SDO_ELEM_INFO_ARRAY(1,1005,2, 1,2,1, 7,2,2, 17,2003,3),
    SDO_ORDINATE_ARRAY(10,50,10,30,50,30,
      50,50,40,60,30,50,20,60,10,50,
      25,35,35,40)));



Konceptuální model objektů SDO_GEOMETRY



Metadata prostorových objektů v Oracle

- Pro správnou funkci Oracle Locator/Spatial je nutno definovat metadata. (popisují, v kterých tabulkách a sloupcích jsou uložena prostorová data a jak vypadá „prostor“, tj. popis dimenzí a souřadnicový systém)
- K metadatům se přistupuje přes pohledy `*_SDO_GEOM_METADATA`. (zápis do `USER_SDO_GEOM_METADATA`, čtení z `ALL_SDO_GEOM_METADATA`)
- Pro každý sloupec `SDO_GEOMETRY` (vrstvu) nutno vložit řádek do `USER_SDO_GEOM_METADATA`.

```
SQL> describe USER_SDO_GEOM_METADATA
Name                 Null?    Type
-------------------- -------- ------------------------------
TABLE_NAME           NOT NULL VARCHAR2(32)
COLUMN_NAME          NOT NULL VARCHAR2(1024)
DIMINFO                       MDSYS.SDO_DIM_ARRAY
SRID                          NUMBER
```

Příklad popisu metadat a validace uložených dat

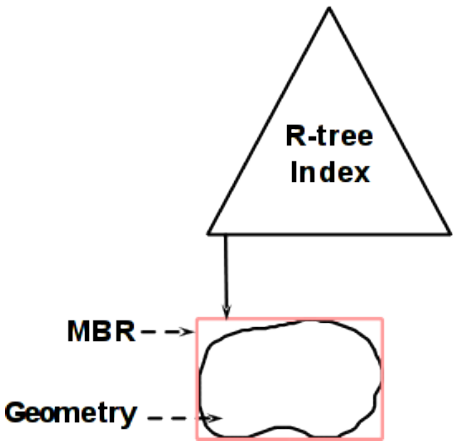
```
INSERT INTO USER_SDO_GEOM_METADATA VALUES (
   'parcela', 'geometrie', SDO_DIM_ARRAY(
     SDO_DIM_ELEMENT('LONGITUDE', -180, 180, 0.5),
     SDO_DIM_ELEMENT('LATITUDE', -90, 90, 0.5)
   ), 8307
);
```

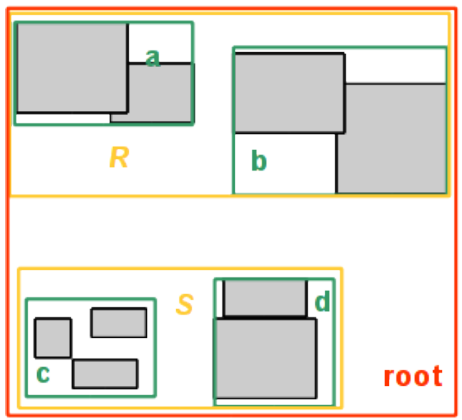
- Definujeme souřadnice jako zeměpisnou délku a šířku v systému WGS 84 (GPS) s přesností na 1/2 stupně.
  (objekty bližší než 1/2 stupně v každé z dimenzí jsou považovány za totožné)
- Objekty v jiném systému souřadnic je nutno vkládat s konverzí.
  (např. `SDO_CS.TRANSFORM(SDO_GEOMETRY(2001, 4156, ...), 8307)`)
- Data lze validovat vzhledem k metadatům a platným `SDO_GTYPE` pomocí metody `ST_IsValid`.

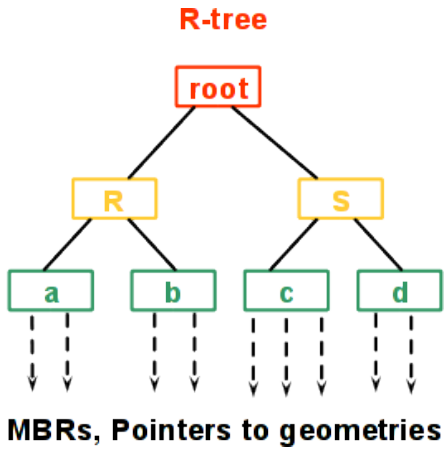
```
SELECT p.cislo, p.geometrie.ST_IsValid() FROM parcela p;
```

Indexování prostorových dat

- Indexy nad prostorovými daty slouží k rychlejšímu vyhledávání.
  (primární filtr funguje až do 4D, sekundární jen do 2D)
- Každá položka indexu aproximuje prostorový objekt pomocí MBR pro 2D nebo MBV pro 3D.
  („Minimum Bounding Rectangle“ a „Minimum Bounding Volume“)
- Indexy jsou uspořádany do R-stromů.
  (cesta upřesňuje MBR/MBV, list odkazuje na geometrii a její min. MBR/MBV)







Tvorba indexu prostorových dat

```
CREATE INDEX parcela_geometrie_sidx ON parcela(geometrie)
  indextype is MDSYS.SPATIAL_INDEX;
```

- Při indexování v prostoru s LRS nutno jeho dimenzi vyjmout.
  (doplnit `PARAMETERS (SDO_INDX_DIMS=2)` u objektů 2D + LRS vzdálenost)
- Data se začnou indexovat (tj. budovat R-strom) po vzniku indexu.
  (možno vyvolat ručně pomocí `ALTER INDEX ...REBUILD`)
- Dokud neskončí indexace (tj. tvorba R-stromu), nemusí být možné index zrušit.
  (v tom případě odstraňovat pomocí `DROP INDEX ...FORCE`)
- Vlastnosti vytvořených indexů jsou přístupné přes pohledy `*_SDO_INDEX_METADATA`.
  (např. vlastnosti vzniklých R-stromů)

Dotazování nad prostorovými daty – Operátory

Operátory

- `SDO_FILTER(`g_1`,`g_2`)`
  (platný při průniku dvou objektů)
- `SDO_RELATE(`g_1`,`g_2`,'MASK=m')`
  (platný při vztahu typu *m* mezi dvěma objekty)
- `SDO_WITHIN_DISTANCE(`g_1`,`g_2`,'DISTANCE=d,UNIT=u')`
  (platný při vzdálenosti nejvýše *d* jednotek *u* mezi dvěma objekty)
- `SDO_NN(`g_1`,`g_2`,'SDO_NUM_RES=n,UNIT=u',r)`
  (platný pro *n* nejbližších sousedů objektu g_2 vybraných z objektů g_1)
- `SDO_NN_DISTANCE(r)`
  (vzdálenost nejbližšího souseda z volání `SDO_NN(...,r)` ve stejném dotazu)

Využívají indexy (nutný v 1. operátoru).
Pokud neuveden typ vztahu (maska), tak tzv. primární filtr (aproximace).
Implicitně transformují mezi souřadnicovými systémy.
Většinou použitelné jen za `WHERE` v podmínce „`= 'TRUE'`“ (nikdy jinak).

Příklady dotazů s operátory

- Najdi všechny parcely s průnikem s parcelou číslo 2:

```
SELECT p1.cislo, p2.cislo FROM parcela p1, parcela p2 WHERE
  (p1.cislo = '2') AND (p1.cislo <> p2.cislo) AND
  SDO_FILTER(p1.geometrie, p2.geometrie) = 'TRUE';
```

- Najdi všechny parcely nacházející se v parcele číslo 2:

```
SELECT p1.cislo, p2.cislo FROM parcela p1, parcela p2 WHERE
  (p1.cislo = '2') AND (p1.cislo <> p2.cislo) AND
  SDO_RELATE(p1.geometrie, p2.geometrie,
    'MASK=inside+coveredby') = 'TRUE';
```

- Najdi 5 nejbližších parcel k parcele číslo 2 a vypiš vzdálenosti:

```
SELECT /*+ ordered */ p1.cislo, p2.cislo,
  SDO_NN_DISTANCE(1) distance
FROM parcela p1, parcela p2 WHERE
  (p1.cislo = '2') AND (p1.cislo <> p2.cislo) AND
  SDO_NN(p2.geometrie, p1.geometrie,
    'SDO_NUM_RES=5 UNIT=meter', 1) = 'TRUE'
ORDER BY distance;
```

Dotazování nad prostorovými daty – Funkce

Funkce

- `SDO_GEOM.RELATE(`g_1`,`g_2`,'m',t)`
  (vrací vztah dvou pobjektů s tolerancí *t*, pokud vztah vyhovuje masce *m*; pro všeobecnou masku použít `determine`)
- . . . a další
  (např. `SDO_GEOM.WITHIN_DISTANCE`, `SDO_GEOM.INTERACT`, atd.)

Nevyužívají indexy (pro menší data) ⇒ tzv. sekundární filtr (přesné).
Nutná shoda souřadnicových systémů.
Použitelné v podmínce za `WHERE` i ve výpisu za `SELECT`.

PŘÍKLAD – Vypiš vztahy ostatních parcel k parcele číslo 2:

```
SELECT p1.cislo, p2.cislo,
  SDO_GEOM.RELATE(p1.geometrie, 'determine', p2.geometrie, 0.5)
FROM parcela p1, parcela p2 WHERE
  (p1.cislo = '2') AND (p1.cislo <> p2.cislo);
```

Prostorové spojení

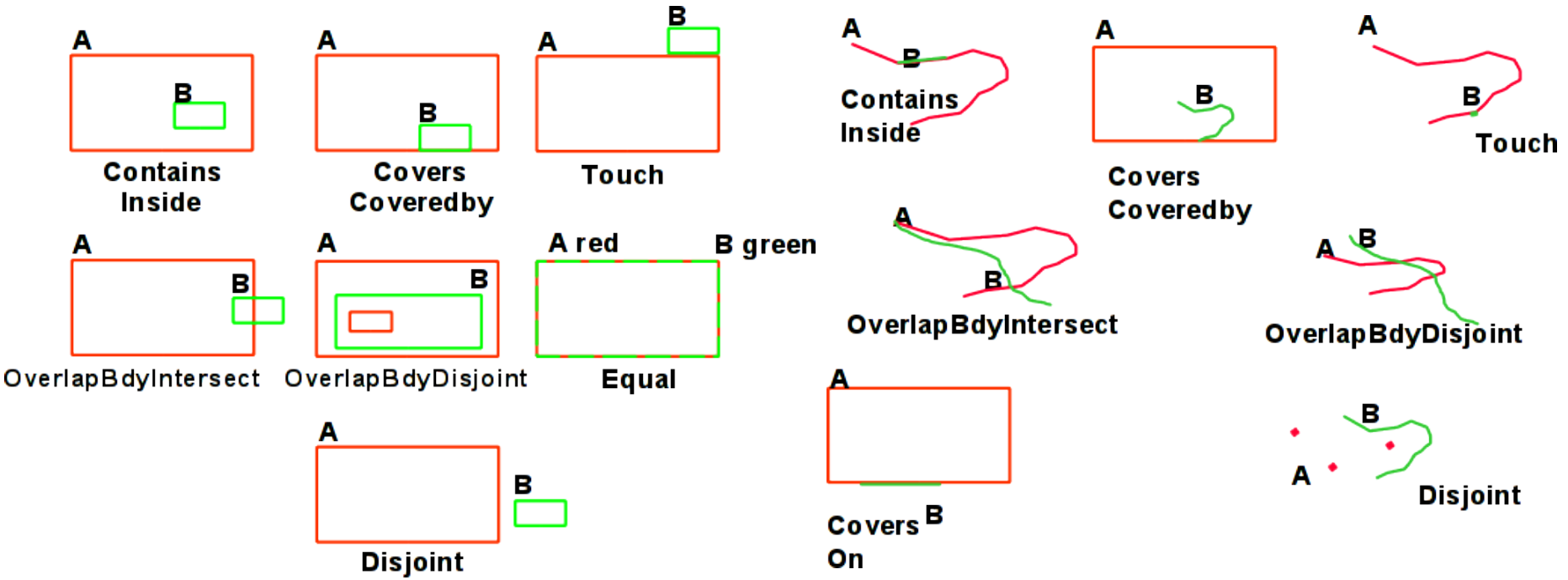
V části `FROM` lze provést prostorové spojení operátorem `SDO_JOIN(tabulka1,sloupec1,tabulka2,sloupec2,'vztah')`.

- Pro porovnání všech nebo většiny objektů ve dvou vrstvách.
  (sloupce musí být indexované, vhodné pro velké objemy zpracovávaných dat)
- Pokud neuveden typ vztahu (např. maska), tak jako primární filtr.
  (využijí se pouze R-stromy indexů a výsledek bude aproximace)

PŘÍKLAD – Vypiš dvojice parcel s nějakým vztahem (tj. ne disjunktní):

```
SELECT p1.cislo, p2.cislo
FROM parcela p1, parcela p2, SDO_JOIN(
    'parcela','geometrie','parcela','geometrie','MASK=anyinteract'
  ) r
WHERE (p1.cislo <> p2.cislo) AND
  (r.rowid1 = p1.rowid) AND (r.rowid2 = p1.rowid);
```

Vztahy prostorových dat



Dotazování nad prost. daty – Analytické operace

Analytické operace

- `SDO_GEOM.SDO_AREA(g,t,u)`
  (vrátí plochu polygonu g v jednotkách u s tolerancí t)
- `SDO_GEOM.SDO_LENGTH(g,t,u)`
  (vrátí obvod polygonu nebo délku čáry g v jednotkách u s tolerancí t)
- `SDO_GEOM.SDO_DISTANCE(g1,g2,t,u)`
  (vrátí nejmenší vzdálenost objektů g_1 a g_2 v jednotkách u s tolerancí t)
- `SDO_GEOM.SDO_BUFFER(g,d,t)`
  (vrátí nový polygon obalující objekt g ve vzdálenosti d s tolerancí t)
- `SDO_GEOM.SDO_CENTROID(g,t)`
  (vrátí nový bod umístěný v těžišti objekty g s tolerancí t)
- `SDO_AGGR_UNION(SDO_GEOM.SDOAGGRTYPE(g,t))`
  (vrátí jeden objekt vzniklý sjednocením objektů g s tolerancí t, tj. agregační fce)
- `SDO_AGGR_CENTROID(SDO_GEOM.SDOAGGRTYPE(g,t))`
  (vrátí jeden bod umístěný v těžišti sjednocení objektů g s tolerancí t, tj. agr. fce)

Export/Import prostorových dat

Nad objektem typu `SDO_GEOMETRY` existují SQL/MM metody:

- `GET_WKB()` – vrací Well Known Binary formát prostorového objektu,
- `GET_WKT()` – vrací Well Known Text fromát prostorového objektu.

WKB/WKT lze použít také při tvorbě objektu `SDO_GEOMETRY` spolu s SRID:

```
INSERT INTO parcela (cislo, geometrie) VALUES (21, SDO_GEOMETRY(
  'POLYGON((146 66, 148 66, 148 68, 146 68, 146 66))', 8307)
);
```

Prostorová data lze též vyexportovat v Geography Markup Language (GML):

```
SELECT cislo, SDO_UTIL.TO_GMLGEOMETRY(geometrie) AS gml
  FROM parcela;
```

Práce s prostorovými objekty v Oracle JDBC I

```
package cz.vutbr.fit.pdb.demo2.spatial;

import java.lang.Exception;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Struct;
import oracle.jdbc.pool.OracleDataSource;
import oracle.spatial.geometry.JGeometry;

class Demo2Spatial {

  public static void main (String args[]) {

    try {
      OracleDataSource ods = new OracleDataSource();
      ods.setURL("jdbc:oracle:thin:@gort.fit.vutbr.cz:1521:dbgort");
      ods.setUser(System.getProperty("login"));
      ods.setPassword(System.getProperty("password"));
      Connection conn = ods.getConnection();

      try {
```

Práce s prostorovými objekty v Oracle JDBC II

```
JGeometry jgeom = null;
Statement stmt = conn.createStatement();
try {

  // reading a geometry from database
  ResultSet rset = stmt.executeQuery(
    "select geometrie from parcela where cislo = 2");
  try {
    if (rset.next()) {
      // convert the Struct into a JGeometry
      Struct obj = (Struct) rset.getObject(1);
      jgeom = JGeometry.loadJS(obj);
    }
  } finally {
    rset.close();
  }
} finally {
  stmt.close();
}

// manipulate the geometry via JGeometry
System.out.println(jgeom.toStringFull());

// writing a geometry back to database
```

Práce s prostorovými objekty v Oracle JDBC III

```
            PreparedStatement pstmt = conn.prepareStatement(
              "update parcela set geometrie=? where cislo = 2");
            try {
              // convert the JGeometry instance to a Struct
              Struct obj = JGeometry.storeJS(conn, jgeom);
              pstmt.setObject(1, obj);
              pstmt.executeUpdate();
            } finally {
              pstmt.close();
            }
          } finally {
            conn.close();
          }
        } catch (SQLException sqlEx) {
          System.err.println("SQLException: " + sqlEx.getMessage());
        } catch (Exception ex) {
          System.err.println("Exception: " + ex.getMessage());
        }
      }

}
```

Překlad a spuštění s Oracle Spatial Java Class Lib.

1. Získat knihovnu Oracle Spatial Java Class Library.
   (z existující instalace Oracle Database SE/SE1/EE soubor `${ORACLE_HOME}/md/jlib/sdoapi.jar`, nebo na `http://www.oracle.com/technetwork/database/enterprise-edition/downloads/` stáhnout „Oracle Database 12c Release 1 SE/SE1/EE“ a z prvního archivu soubor `./database/stage/Components/oracle.sdo.locator/*/DataFiles/filegroup4.jar:./md/jlib/sdoapi.jar`)
2. Knihovnu přidat do `CLASSPATH` pro překlad a spuštění.
   (`javac -classpath ...:./lib/sdoapi.jar...`)

Předchozí příklad:

```
$ javac -classpath .:./lib/ojdbc7.jar:./lib/sdoapi.jar \
    cz/vutbr/fit/pdb/demo2/spatial/Demo2Spatial.java
$ java -classpath .:./lib/ojdbc7.jar:./lib/sdoapi.jar \
   -Dlogin=*** -Dpassword=*** cz.vutbr.fit.pdb.demo2.spatial.Demo2Spatial
JGeometry (gtype=1, dim=2, srid=8307,
 ElemInfo(1,1,1),
 Ordinates(-60.9863151932323,67.9988704665108
))
```

Literatura k prostorovým databázím

- Ravi Kothuri, Albert Godfrind, Euro Beinat: **Pro Oracle Spatial for Oracle Database 11g**. ISBN 1590598997

  `http://www.apress.com/book/view/9781590598993`
- **Oracle Locator**.

  `http://docs.oracle.com/database/121/SPATL/sdo_locator.htm`
- **Oracle Spatial Java API**.

  `http://docs.oracle.com/database/121/SPAJV/toc.htm`
- Jiří Činčura: **MS SQL 2008 – prostorová data**. Databázový svět.

  `http://www.dbsvet.cz/view.php?cisloclanku=2009101201`

Oracle jako XML databáze

- V klasické relační databázi jsou dva způsoby uložení XML dokumentu:
  a) rozparsovat XML v aplikaci a do db. uložit pozice (cestu v XML DOM) a hodnoty (text) jednotlivých elementů a atributů,
  b) uložit celé XML do db. jako binární data (např. `CBLOB`).

  Toto nepodporuje přímé dotazování a manipulaci s XML na úrovni db.

- Oracle XML DB nabízí pro XML dokument datový typ `XMLType`:
  - podporuje W3C XML datový model a standardy XPath, SQL/XML,
  - spojuje výhody relačních databází a technologie XML,
    (tzv. „XML/SQL dualita“, XML přístup k SQL datům a SQL nad XML daty)
  - poskytuje indexy nad XML dokumentem,
    (B*stromy struktury, text-indexing hodnot, XML-text index pro XPath)
  - provázanost s API databáze.
    (podpora přístupu ke XML na aplikační úrovni pomocí JDBC)

Použití `XMLType` v Oracle XML DB

Hodnoty typu `XMLType` mohou být uloženy:

- ve sloupcích tabulky, pokud potřebujeme další (relační) data

```
CREATE TABLE sbirka_knih (
  id_sbirky INTEGER PRIMARY KEY,
  obsah XMLType
);
```

- v tabulce celé, pokud je potřeba pouze XML dokument

```
CREATE TABLE obsah_sbirky_knih OF XMLType;
```

Vložení záznamu s `XMLType` do tabulky

- pomocí API databáze, např. před JDBC v Javě:

```
public void doInsert(Connection conn, Document doc) throws Exception {
  String SQLTEXT = "INSERT INTO purchaseorder VALUES (?)";
  XMLType xml = XMLType.createXML(conn,doc);
  OraclePreparedStatement sqlStatement =
    (OraclePreparedStatement) conn.prepareStatement(SQLTEXT);
  sqlStatement.setObject(1,xml);
  sqlStatement.executeUpdate();
}
```

- přímo v SQL příkazu `INSERT`:

```
INSERT INTO sbirka_knih(id_sbirky,obsah) VALUES (1, XMLType(
  '<?xml version="1.0" encoding="UTF-8" ?>
  <books xmlns="http://www.fit.vutbr.cz/.../obsah_sbirky_knih">
    <book publisher="IDG books" on-loan="Sanjay" >
      <title>XML Bible</title>
      <author>Elliotte Rusty Harold</author>
    </book>
    <book publisher="QUE">
      <title>XML By Example</title>
      <author>Benoit Marchal</author>
    </book>
  </books>'));
```

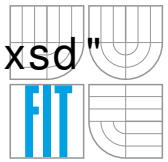
Definice XML schéma pro záznamy typu `XMLType`

Hodnoty záznamů je možno validovat proti XML Schema:

1. zaregistrujeme nové XML Schema pomocí
   `dbms_xmlschema.registerSchema(schemaurl,schemadoc)`
   - toto volání parsuje a validuje dané XML Schema,
   - vytvoří odpovídající položky v Oracle Data Dictionary,
   - definuje SQL objekty pro elementy „complexType“ v XML schema,
   - vytvoří tabulky `XMLType` všem globálním elementům XML schema.
2. přiřadíme schéma sloupci nebo tabulce typu `XMLType`

```
CREATE TABLE sbirka_knih (
   id_sbirky INTEGER PRIMARY KEY,
   obsah XMLType
) XMLType COLUMN obsah
   XMLSCHEMA "http://www.fit.vutbr.cz/.../obsah_sbirky_knih.xsd"
   ELEMENT "books";

CREATE TABLE obsah_sbirky_knih OF XMLType
   XMLSCHEMA "http://www.fit.vutbr.cz/.../obsah_sbirky_knih.xsd"
   ELEMENT "books";
```



Příklad zrušení a zavedení registrace XML schéma

```
DBMS_XMLSCHEMA.deleteSchema(
   schemaURL => 'http://www.fit.vutbr.cz/.../obsah_sbirky_knih.xsd',
   delete_option => DBMS_XMLSCHEMA.DELETE_CASCADE_FORCE
);
DBMS_XMLSCHEMA.registerSchema(
   schemaURL => 'http://www.fit.vutbr.cz/.../obsah_sbirky_knih.xsd',
   schemaDOC => '<?xml version="1.0" encoding="UTF-8" ?>
   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.fit.vutbr.cz/.../obsah_sbirky_knih"
      xmlns:bk="http://www.fit.vutbr.cz/.../obsah_sbirky_knih"
      elementFormDefault="qualified">
      <xs:element name="books"><xs:complexType>
        <xs:sequence>
          <xs:element name="book" type="bk:bookType" maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="collection" type="xs:string" use="required" />
      </xs:complexType></xs:element>
      <xs:complexType name="bookType">
        <xs:sequence>
          <xs:element name="title" type="xs:string" />
          <xs:element name="author" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="publisher" type="xs:string" />
        <xs:attribute name="on-loan" type="xs:string" use="optional" />
      </xs:complexType>
   </xs:schema>');
```

Registrace a přiřazení XML schéma – důsledky

Přiřazení XML Schema k hodnotám typu `XMLType` umožní:

- omezit XML dokumenty pouze na validní (nejen „well-formed“),
- definovat referenční integritu na hodnoty elementů/atributů,
- vytvářet další omezení, např. `UNIQUE` pro uzly v XML, atd.

PŘÍKLAD –
Jedinečnost hodnot atributu `collection` elementu `books` v tabulce:

```
CREATE UNIQUE INDEX sbirka_knih_obsah_col_idx
ON sbirka_knih (
extractValue(obsah, '/books/@collection')
);
```

Dotazování XML dat v Oracle XML DB pomocí XPath

- `existsNode(xml,xpath)`
  (test na obsah uzlu definovaného XPath výrazem v daném dokumentu, vrací 0/1)
- `extract(xml,xpath)`
  (vrací podstromy elementů dle XPath v daném dokumentu, výsledek je `XMLType`)
- `extractValue(xml,xpath)`
  (vrací hodnotu textového uzlu/atributu dle XPath v daném dokumentu, tj. SQL typ)
- `value(xml)`
  (vrací celý XML dokument, výsledek je `XMLType`)

Pozn.: XPath výraz může mít problémy s elementy s definovaným jmenným prostorem.

PŘÍKLAD –
Vypiš uzel na 3. stupni 1. větve DOM stromu jako element a jako text:

```
SELECT id_sbirky,
       extract(obsah, '/*[1]/*[1]/*[1]') as element,
       extractValue(obsah, '/*[1]/*[1]/*[1]') as text
FROM sbirka_knih;
```

Aktualizace XML dat v Oracle XML DB

- `updateXML(xml,xpath,value)`
  (upraví uzel definovaný XPath výrazem v daném dokumentu na danou hodnotu)

Takto lze nahradit libovolný uzel DOM stromu (podstrom elementu, textový obsah elementu i atributu).

PŘÍKLAD –
Změň textový uzel v elementu na 3. stupni 1. větve DOM stromu:

```
UPDATE sbirka_knih
SET obsah = updateXML(obsah, '/*[1]/*[1]/*[1]/text()', 'New XML Bible')
WHERE id_sbirky = 1;
```

Příklad práce s XML přes Oracle JDBC I

```
package cz.vutbr.fit.pdb.demo2.xml;

import java.lang.Exception;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import oracle.jdbc.pool.OracleDataSource;
import oracle.xdb.XMLType;
import oracle.xml.parser.v2.XMLDocument;

class Demo2XML {

  public static void main (String args[]) {

    try {
      OracleDataSource ods = new OracleDataSource();
      ods.setURL("jdbc:oracle:thin:@gort.fit.vutbr.cz:1521:dbgort");
      ods.setUser(System.getProperty("login"));
      ods.setPassword(System.getProperty("password"));
      Connection conn = ods.getConnection();

      try {
```

Příklad práce s XML přes Oracle JDBC II

```
XMLDocument doc = null;
Statement stmt = conn.createStatement();
try {

  // reading a XML DOM from database
  ResultSet rset = stmt.executeQuery(
    "SELECT extract(obsah, '/*[1]/*[1]/*[1]') " +
    "FROM sbirka_knih WHERE id_sbirky = 1");
  try {
    if (rset.next()) {
      // convert the XMLType object into a XMLDocument
      XMLType xml = (XMLType) rset.getObject(1);
      doc = (XMLDocument) xml.getDocument();
    }
  } finally {
    rset.close();
  }
} finally {
  stmt.close();
}

// manipulate the DOM
doc.print(System.out);
```

Příklad práce s XML přes Oracle JDBC III

```
            // writing a DOM back to database
            PreparedStatement pstmt = conn.prepareStatement(
              "UPDATE sbirka_knih SET obsah = " +
              "updateXML(obsah, '/*[1]/*[1]/*[1]', ?) WHERE id_sbirky = 1");
            try {
              // convert the DOM to a XMLType instance
              XMLType xml = XMLType.createXML(conn, doc);
              pstmt.setObject(1, xml);
              pstmt.executeUpdate();
            } finally {
              pstmt.close();
            }
          } finally {
            conn.close();
          }
        } catch (Exception ex) {
          System.err.println("Exception: " + ex.getMessage());
          ex.printStackTrace();
        }
      }

}
```

Překlad a spuštění příkladu s XML

1. Získat knihovny Oracle XML DB.
   (ze stránek `http://www.oracle.com/technetwork/database/features/xmldb/java-utilsoft-087831.html` stáhnout „Oracle XML Developer's Kit (XDK) Java“ a z archivu získat soubory `./lib/xdb.jar` a `./lib/xmlparserv2.jar`)
2. Knihovny přidat do `CLASSPATH` pro překlad a spuštění.
   (`javac -classpath ...:./lib/xdb.jar:./lib/xmlparserv2.jar...`)

Předchozí příklad:

```
$ javac -classpath .:./lib/ojdbc7.jar:./lib/xdb.jar:./lib/xmlparserv2.jar \
    cz/vutbr/fit/pdb/demo2/xml/Demo2XML.java
$ java -classpath .:./lib/ojdbc7.jar:./lib/xdb.jar:./lib/xmlparserv2.jar \
    -Dlogin=*** -Dpassword=*** cz.vutbr.fit.pdb.demo2.xml.Demo2XML
<title xmlns="http://www.fit.vutbr.cz/~rychly/pdb/cv2/xml/obsah_sbirky_knih">
  New XML Bible
</title>
```

Literatura k Oracle XML DB

- Mark V. Scardina, Ben Chang, Jinyu Wang: **Oracle Database 10g: XML & SQL: design, build & manage XML applications in Java, C, C++ & PL/SQL**. ISBN 0072229527

  `http://books.google.com/books?isbn=0072229527`, `http://www.mhprofessional.com/downloads/products/0072229527/0072229527_code.zip`
- **Oracle XML DB home**.

  `http://www.oracle.com/technetwork/database/database-technologies/xmldb/overview/index.html`
- **Oracle XML DB Developer's Guide**.

  `http://docs.oracle.com/database/121/ADXDB/toc.htm`
- **Oracle XML Developer's Kit Programmer's Guide**.

  `http://docs.oracle.com/database/121/ADXDK/toc.htm`
- **Oracle Database XML Java API Reference**

  `http://docs.oracle.com/database/121/JAXML/toc.htm`

Děkuji za pozornost.

Otázky? Diskuze?