Řešení správy rastrových dat s využitím databáze Oracle 9i a MicroStation v8

Ing. Stanislav Šumbera, Ph.D.
Mendelova zemědělská a lesnická univerzita Brno
Zemědělská 3, 613 00
www: http://sumbera.com
E – mail ambrosa@mendelu.cz

Abstract

Raster data such as satellite images, air photos, scanned topographical maps or other types of imagery are becoming increasingly popular in GIS. It has been estimated that these data may represents more than 90% of the average GIS data holding by volume. One of the most used database to manage all kinds of data across the enterprise is Oracle. Oracle 9i brings its GIS technologies like Spatial option, interMedia or Database Workspace Manager for effective database storing of geographical-related data.
In this article, Oracle GIS technologies are used to build up an open-source prototype called "Image Storage" for storing rasters in Oracle9i database and displaying query results with help of MicroStation v8 Raster Manager.

Abstrakt

Rastrová data (ať už mapové podklady, DMT modely, nebo snímky DPZ) tvoří až 90% průměrného množství GIS dat co se týče objemu. Efektivní správa rastrů proto představuje fundamentální funkčnost pro geografické systémy. Přesto je tomuto tématu stále věnována poměrně malá pozornost. Jednou z nejpoužívanějších databází pro správu heterogenních data je databáze Oracle. Verze Oracle 9i obsahuje vylepšené GIS technologie jako například Spatial, interMedia nebo Workspace Management pro efektivní ukládání geografických dat.
V tomto příspěvku jsou tyto technologie využity pro sestavení prototypu open-source aplikace „Image Storage“, která do databáze Oracle rastry ukládá a v prostředí MicroStation v8 tyto rastry zobrazuje.

Úvod

Systémy pro správu geografických rastrů můžeme rozdělit do třech typů (generací) :

1. Rastry jsou ukládány v proprietárním nebo publikovaném formátu do souborového systému.
2. Rastry jsou ukládány do proprietární rastrové "databáze" (Mr.SID, ECW).

3. Rastry jsou ukládány v publikovaném formátu do obecné databáze s prostorovým klíčem. Zakomponování výkonných kompresních algoritmů. např. RasDB, SDE, ImageStor apod.

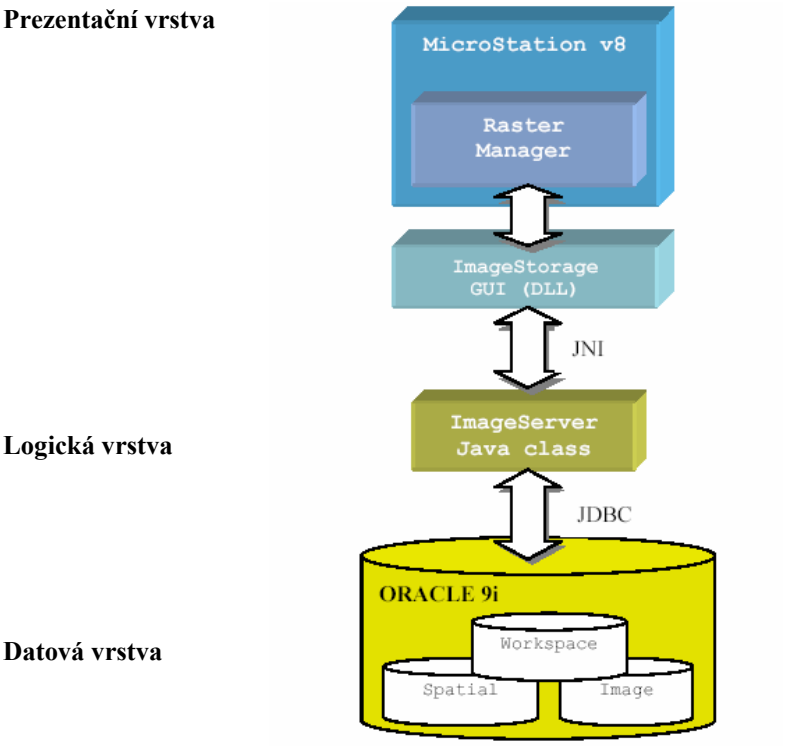
Posledně uvedeným typ ukládání rastrů je aplikován v tomto příspěvku pro databázi Oracle 9i, která obsahuje řadu zajímavých vlastností pro správu prostorových dat. Namísto teoretických informací o možnostech této databáze je tento příspěvek zaměřen prakticky, tzn. byl vytvořen prototyp aplikace, která demonstruje práci s rastry v databázi Oracle 9i a možnosti zobrazování rastrů prostřednictvím MicroStation v8. Tento prototyp je volně k dispozici pro studijní a výzkumné účely.

Hlavní rysy aplikace Image Storage

Aplikace Image Storage pro management snímků umožňuje moderním způsobem ukládat, vyhledávat a vybírat rastrová data z databáze Oracle v jednom ze standardních rastrových formátů prostřednictvím Oracle Image Cartridge. Prostorová indexace a vyhledávání rastrů je založena na technologii Oracle Spatial. Rastrová data jsou spravována v tabulkách, nad kterými lze provádět dlouhé transakce prostřednictvím Oracle Workspace Manageru - tím je umožněno, aby se daný rastr nacházel v různých rozpracovaných stadiích (stavech) pro účely zpracování snímku. Zobrazení rastrů je umožněno v grafickém prostředí systému MicroStation.

Architektura

Datová vrstva aplikace je reprezentována databází Oracle, konkrétně entitami vytvořenými pro management rastrů. Na logické vrstvě využívá Image Storage aplikačního rozhraní Javy pro komunikaci s Oraclem. Na úrovni prezentační je GUI aplikace vytvořeno v jazyku VC++ a s využitím tříd MFC. Pro zobrazování rastrů je využito knihovny Raster Manageru z MicroStation, viz obr1.



Obr. 1: Přehled architektury Image Storage.

Datová vrstva:

Pro ukládání rastrových dat do databáze, jejich prostorovou lokalizaci a možnost managementu prostřednictvím dlouhých transakcí byly využity tři GIS technologie databázového serveru Oracle :

1. Image Cartridge

umožňuje ukládat, získávat a konvertovat rastrová data prostřednictvím objektového datového typu (ODT). Snímky se ukládají do databáze buď ve formě BLOB objektů - (binary large objects), nebo jako odkaz na souborový systém (objekty BFILE). Image Cartridge bylo využito jako základní technologie pro ukládání rastrů do databáze. Bližší specifikace technologie viz. firemní dokumentace ORACLE.

2. Spatial Cartridge:

je představován souborem SQL funkcí a schémat, která umožňují ukládání, získávání, dotazování a aktualizaci prostorových objektů v databázi Oracle 9i . Tato technologie byla využita pro prostorovou lokalizaci rastru v databázi.

Objekty z těchto dvou cartridge jsou využity v tabulce aplikace *img*_**storage** (předpona je volitelná část), která obsahuje mimo jiné objekt *ordimage*, do kterého se ukládá rastr v podobě BLOBu a objekt *sdo_geometry*, ve kterém jsou uloženy geometrické souřadnice. Tabulka dále obsahuje primární klíč a název rastru:

```
create table  img_storage (ID     number PRIMARY KEY not null,
                           NAME   varchar2 (256),
                           IMAGE  ordsys.ordimage,
                           MBR    mdsys.sdo_geometry);
```

Pro tabulku je založen index a sekvence pro generování primárního klíče :

```
create index  img_idx on  img_storage(MBR)
indextype is mdsys.spatial_index PARAMETERS('SDO_LEVEL = 8');

create sequence img_seq increment by 1;
```

Příklad prostorového vyhledávání rastrů:

A. Prostorový dotaz bez využití Spatial cartridge :

```
select name from img_storage where not
((x_min < -765900000 AND x_max < -765900000) or
(x_min > -765600000 AND x_max > -765600000) or
(y_min > -966000000 AND y_max > -966000000) or
(y_min < -966100000 AND y_max < -966100000))
```

B. Tentýž prostorový dotaz s využitím Spatial Cartridge :

```
select a.id from img_storage a
where sdo_filter(a.mbr,mdsys.sdo_geometry(2003,NULL,NULL,
mdsys.sdo_elem_info_array(1,1003,3),
mdsys.sdo_ordinate_array(-765900000,-966100000,-765600000,-966000000)),
'querytype=window')= 'TRUE'
```

3. Workspace Management (WM)

Technologie pro verzování tabulek. WM umožňuje, aby databáze udržovala rozdílné verze stejného záznamu (tj. řádku v tabulce) v jednom nebo více workspaces (prostředí, stavů). Uživatelé databáze potom mohou měnit tyto verze nezávisle. Tento přístup verzování v databázi má dvojí výhodu:

1. Verzování zdokonaluje konkurenční přístup k datům v databázi. Bez verzování jsou příslušné záznamy v případě konkurenčního přístupu uzamčeny.
2. Verzování umožňuje analýzy různých stavů dat současně, protože každá z těchto analýz pracuje nad separátní verzí dat.

Jednotkou verzování je databázová tabulka. Běžná tabulka může být transformována do verzovací tabulky, což znamená, že veškeré řádky v tabulce mohou podporovat různé verze dat. Z pohledu uživatele se tyto verze jeví jako jedna a táž tabulka, která se nachází v určitém workspace (stavu). Stav je virtuálním prostředím, které mohou sdílet uživatelé, aby do verzovaných tabulek prováděli změny (ORACLE 2001). Tyto Stavy mají mezi sebou hierarchickou strukturu: Každý Stav má svého předka a z každého Stavu lze odvodit potomka. Viz obr. 2.

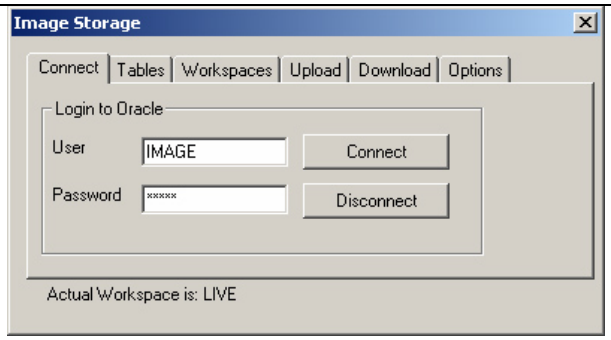
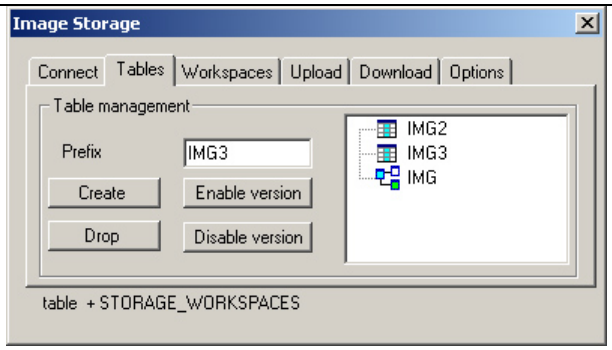
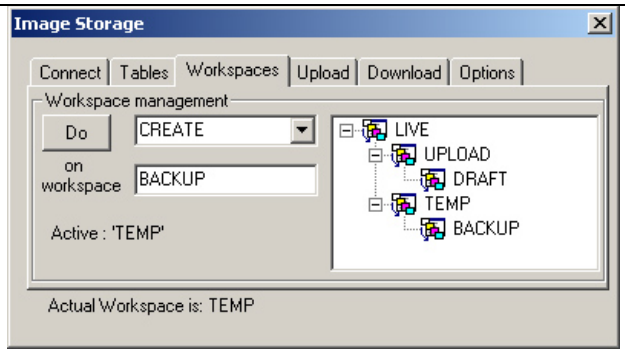
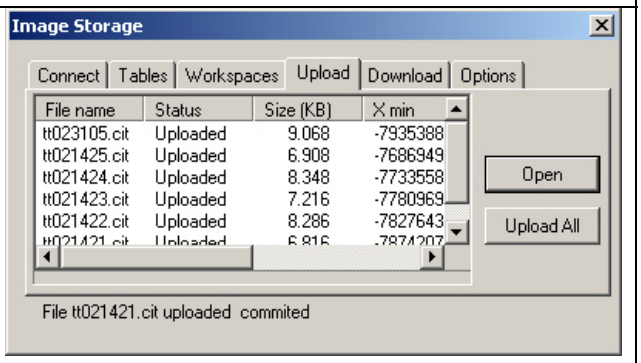
Obr. 2: Hierarchická struktura Stavů. Základní Stav je vždy LIVE, z něj je v tomto případě odvozen Stav pro upload rastrových dat s názvem UPLOAD. Pro dočasné změny v rastru byl vytvořen Stav DRAFT, TEMP a BACKUP.

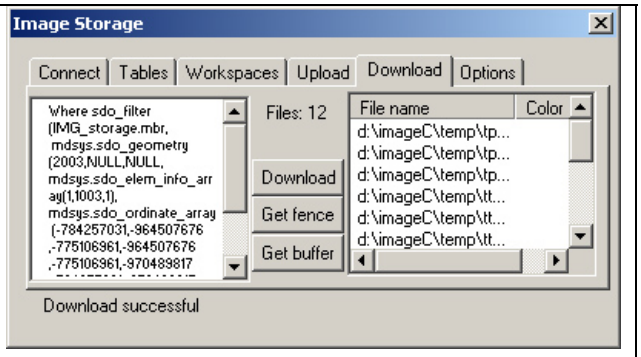
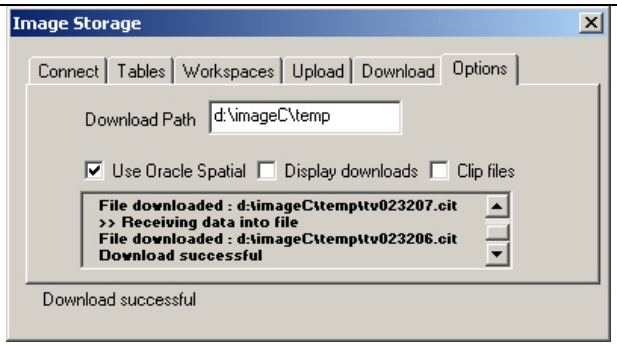
Logická vrstva

Logická vrstva aplikace je realizovaná prostřednictvím jazyka Javy. Zvolení tohoto jazyka bylo podmíněno především javovským rozhraním Oracle pro práci s rastry. Komunikace s Oracle databází se uskutečňuje prostřednictvím JDBC - Java Database Connectivity. Příkazy jsou vyvolávány statickým nebo dynamickým SQL. V aplikaci Image Storage je komunikace mezi logickou vrstvou a vrstvou prezentační řešeno pomocí JNI - Java Native Interface (viz dodatek)

Uživatelská vrstva

Uživatelské rozhraní bylo napsáno v C++ s využitím MFC - Microsoft Foundation Classes a to z důvodu snadnější komunikace mezi kódem C++ a Javou. Kvůli zobrazování rastrů v prostředí MicroStation bylo uživatelské rozhraní napsáno jako DLL knihovna. Tato knihovna je v prostředí MicroStation zavedena pomocí aplikace napsané v MDL (MicroStation Development Language). Pokud ovšem zobrazování není podmínkou (např. je třeba pouze rastry do databáze ukládat) je možné spustit aplikaci pomocí samostatného EXE souboru. Kód uživatelského rozhraní (C++) komunikuje s logickou vrstvou (Java) prostřednictvím JNI - Java Native Interface. Přehled GUI je uveden v následující tabulce.

Popis funkčnosti	*Uživatelské rozhraní*
<u>Záložka pro připojení do databáze</u> **USER -** název uživatele **PASSWORD** - heslo pro uživatele **CONNECT** - přihlášení do databáze **DISCONNECT** - odhlášení z databáze	
<u>Záložka pro management tabulek</u> **CREATE** - založení tabulky **DROP** - zrušení tabulky **ENABLE VERSION** - nastavení verzování pro tabulku **DISABLE VERSION** - zrušení verzování pro tabulku	
<u>Záložka pro management Workspaces</u> **DO -** vykonání příslušné operace nad Workspace Aktivní Workspace se nastaví kliknutím na požadované jméno Workspace	
<u>Záložka pro upload rastrů do databáze</u> **OPEN -** otevře se dialogové okno pro výběr souborů, které se budou ukládat do databáze **UPLOAD ALL -** spustí se ukládání (upload)	

Záložka pro download rastrů z databáze **levé okno**: zde se specifikuje SQL podmínka pro výběr rastrů např. prostorové omezení **GET FENCE** - podmínka se nastaví podle pozice ohrady v MicroStation **DOWNLOAD** - provede se download	
Záložka pro nastavení vlastnostíaplikace **USE ORACLE SPATIAL :** použije se vyhledávání podle prostorového klíče. **DISPLAY DOWNLOADS** : v prostředí MicroStation se rastry zobrazí **CLIP FILES :** v prostředí MicroStation se rastry ořežou dle ohrady.	

Závěr

Příspěvek si kladl za cíl ukázat praktickou cestu jak ukládat rastrová data do databáze Oracle 9i s využitím nejmodernějších technologií jakými jsou bezesporu Workspace Manager, Image Cartridge a Spatial Cartridge. Uvedené příklady i zdrojový kód jsou k dispozici na adrese www.sumbera.com

Dodatky

Technologie komunikace mezi logickou a prezentační vrstvou - JNI

JNI je technologie, která umožňuje kombinovat kód napsaný v Javě s kódem nativním. JNI je specifikováno korporací SUN MICROSYSTEMS. JNI technologie umožňuje časově náročnější operace nebo uživatelské rozhraní realizovat v nativním kódu pro daný operační systém, resp. naopak, část kódu realizovat v Javě. Alternativní možností k JNI je technologie J/Direct firmy Microsoft, jejíž hlavní výhodou je rychlost komunikace. Nativní kód je v prostředí Win32 reprezentovaný DLL knihovnou nebo EXE souborem. JNI umožňuje vyvolávat exportované funkce z DLL knihovny a naopak.

Prostřednictvím JNI je umožněno :

1. Spustit metodu třídy napsané v Javě
2. Spustit nativní funkci umístěnou v DLL
3. Manipulovat s javovskými objekty
4. Operovat s výjimkami
5. Zavádět javovské třídy

Přístup k nativní funkci z Javy

Příklad využití knihovny DLL a exportované funkce uvádí následující příklad:

```
class JavaJni{
static {
    System.loadLibrary("nativeLib"); // natáhne se knihovna s názvem
nativeLib.dll
   }
  public native void callDllFunc(short limit); // deklarace nativní funkce
...
}
```

Pro vytvoření implementace této nativní metody v DLL knihovně lze využít nástroj **javah.exe s parametrem -jni** a názvem třídy, ve které se deklarace nativní metody vyskytuje. Výsledkem bude deklarace nativní funkce v hlavičkovém souboru (přípona .h) pro nativní kód. Například při zadání : **javah – jni JavaJni** se vytvoří soubor "**JavaJni.h**" který bude obsahovat prototyp v C++ pro funkci **callDllFunc**:

```
#include <jni.h>
extern "C" {
JNIEXPORT jint JNICALL
Java_JavaJni_callDllFunc  (JNIEnv *jniEnv, jobject jObject, jshort param); //
deklarace nativní funkce
}
```

kde *jniEnv* je ukazatel na běžící virtuální stroj Javy, *jObject* představuje v tomto případě instanci objektu JavaJni a *param* je parametr funkce.

Přístup k metodě z Javy z nativního kódu

Rozšíříme výše uvedenou třídu JavaJni :

```
public class JavaJni {
   static JavaJni     javaObj  = new JavaJni();
public int onNativeCall(short limit){ // metoda volaná z nativního kódu...
       return true; }
}
```

Pro volání z nativního kódu je třeba provést následující kroky:

1. Obdržet ukazatel na běžící virtuální stroj Javy
2. Získat ukazatel na JNI rozhraní

3. Získat třídu, kde je umístěna metoda
4. Získat identifikátory metody a objektu podle signatury a názvu metody vrácené z utility javap
5. Získat požadované instance objektu
6. Zavolat nestatické javovské metody pomocí příslušné funkce

Příklad implementace v nativním kódu je uveden v následujícím příkladu:

```
int  JNICALL dll_javaMethodCall(short limit)
{
 JavaVM            *jvm;       /* označuje Java viturální stroj */
 JNIEnv             *env;      /*  ukazatel na JNI              */
 jsize               nVMs;     /*  počet vytvořených JVM        */

/* 1.*/ jint res       = JNI_GetCreatedJavaVMs(&jvm,(jsize)1,&nVMs);
/* 2.*/ res            = jvm->AttachCurrentThread(&env,NULL);
/* 3.*/ jclass    cls  = env->FindClass("JavaJni");
/* 4.*/ jmethodID mid  = env->GetMethodID(cls, "onNativeCall", "(S)I");
        jfieldID  fid  = env->GetStaticFieldID(cls, "javaObj", "LJavaJni;");
/* 5.*/ jobject   obj  = env->GetStaticObjectField(cls, fid);
/* 6.*/ return  env->CallIntMethod(obj, mid,limit);
}
```

Příklad downloadu rastru z databáze

```
/*-----------------------------------------------------------------------------*/
  public int downloadImage
  (
    String pre,                          // table prefix
    String where,                        // where statement for SQL
    String path                          // download path
  )
 /*-----------------------------------------------------------------------------*/
 // downloads all images specified by where condition
  {

    String     select =   "select IMAGE,NAME from "+pre+"_storage " + where;
    int        counter = 0;

    try{
      int index1 = 0;
      Statement s1 = con.createStatement();
      OracleResultSet rs1 =(OracleResultSet)s1.executeQuery(select);
      while(rs1.next()){
        OrdImage imgObj   = (OrdImage)
                            rs1.getCustomDatum(1,OrdImage.getFactory());
        String   fileName = rs1.getString(2);
        try {
          counter++;
          int index =  fileName.lastIndexOf("\\");
          String name = fileName.substring(index+1);
          String fullName = path + "\\"+name;
          try{
           sendLog(">> Receiving data into file \n");
           imgObj.getDataInFile(fullName);
          }
```

```
        catch (Exception e){ sendLog(">> " + e +"\n"); }

        loadFile(fullName,0); //  native call
        sendLog("File downloaded : "+fullName+"\n");
      }
      catch (Exception e) { sendLog(">> " + e +"\n"); }

    }
    rs1.close();
    s1.close();
  }
  catch(Exception e) { sendLog(">> " + e); }

  sendLog("Download successful \n");
  return counter;
}
```

Příklad uploadu rastru do databáze

```
/*--------------------------------------------------------------------------*/
  public void uploadImage
  (
    String pre,                          // Image table prefix
    String FileName,                     // Raster File Name to upload
    String sqlInsert,                    // inserting SQL statement
    String seq,                          // sequence number (ID)
  )
  /*--------------------------------------------------------------------------*/
  // upload one raster file into database
  {
   try {

    String      sql;
    OraclePreparedStatement stmt;
    String      name = FileName.substring(FileName.lastIndexOf("\\")+1);

    sql = "insert into "+pre+"_storage values "+
            "("+ seq +",'"+ name +"',"+ sqlInsert+ ")";
    stmt =   (OraclePreparedStatement)con.prepareCall(sql);

    sendLog(">> Executing insert "+"\n");
    try{
    stmt.execute();
    }
    catch (SQLException e) { sendLog(">> " + e); }
    stmt.close();

    sql ="select IMAGE from "+pre+"_storage where NAME = '"+
             +name+"' for update";
    Statement s1 = con.createStatement();
    OracleResultSet rs1 =  (OracleResultSet)s1.executeQuery(sql);
     while(rs1.next())
      {
        OrdImage imgObj = (OrdImage)
                          rs1.getCustomDatum(1, OrdImage.getFactory());
        sendLog(">> loading data from file ... \n");
        imgObj.loadDataFromFile(FileName);

        sql ="update "+pre+"_storage set image = ? where NAME = '"+ name+"'";
        sendLog(">> Executing update \n");
        stmt = (OraclePreparedStatement) con.prepareCall(sql);
```

```
      stmt.setCustomDatum(1,imgObj);
      try{
      stmt.execute();
      }
      catch (SQLException e) { sendLog(">> " + e); }
      stmt.close();

      sendLog("File "+name+" uploaded \n");
      stmt.close();
    }

   stmt.close() ;
  }
  catch(Exception e) {
    sendLog(">> " + e+ "\n");
  }
}
```