2010 Esri 中国区域用户大会

海量空间数据库实施策略

吴泳锋 刘锋

内容

- 序言
- 空间数据库设计
- 矢量数据实施策略
- 栅格数据的实施策略

内容

- 序言
  - 什么是GDB
  - 什么算是海量数据
  - 谁能胜任
- 空间数据库设计
- 矢量数据实施策略
- 栅格数据的实施策略

什么是GDB

- 用来存储，查询，管理空间数据的数据库或者文件
  - Personal GDB(access)
  - File GDB
  - RDBMS
    - DB2
    - Informix
    - Oracle
    - PostgreSQL
    - SQL Server

什么数据算是海量？

- 数据量大？

- 1T Layer

Feature数量	平均Feature大小	总大小
500	2G	1T
5000	200M	
50000	20M	
500000	2M	
5000000	200K	
50000000	20K	
500000000	2k	

Why?

- 应用
- 管理
- 维护

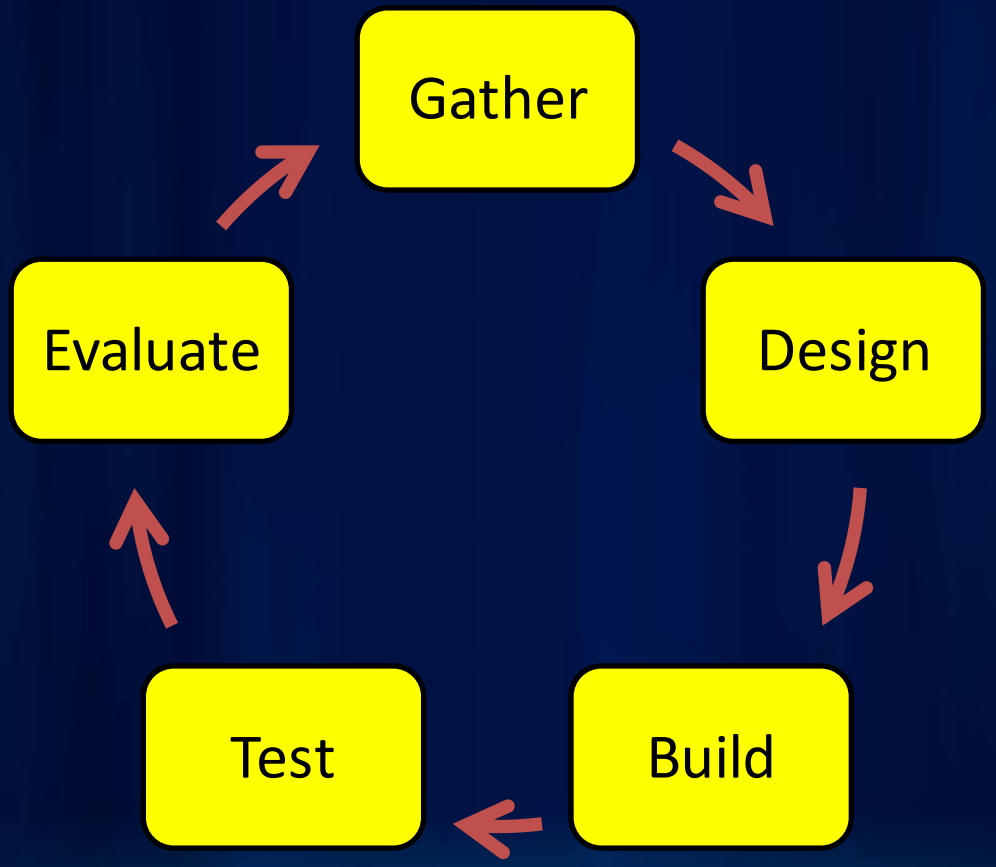
谁能胜任

- Personal GDB(access)
- File GDB
- RDBMS
  - DB2
  - Informix
  - Oracle
  - PostgreSQL
  - SQL Server

内容

- 序言
- 空间数据库设计
  - 设计流程
  - 数据建模
  - 数据组织
- 矢量数据实施策略
- 栅格数据的实施策略

设计流程



数据建模

- 主要任务

  逻辑模型

  物理模型

数据建模

- 目标

  建立一套适合于自己应用的数据结构，包括表，关系，元信息，拓扑规则等

- 工具
  - Visio 2002 & 2003
  - ESRI Tools
    - ArcGIS Diagrammer 9.2 & 9.3
    - GDB Xray
    - Zieler’s GDB Diagrammer 8.x & 9x
    - XML Validator
    - Geodatabase Reporter .Net 8.x & 9.x
  - Rational Rose

主要应用场景

- 软件架构
  - C/S
  - B/S
  - C/S & B/S

- 业务性质
  - OLTP
  - OLAP
  - OLTP/OLAP

内容

- 序言
- 空间数据库设计
- 矢量数据实施策略
  - 硬件策略
  - 软件策略(OS, DBMS, ARCGIS)
- 栅格数据的实施策略

硬件设备策略

- 数据库服务器和应用服务器

- CPU
  频率，核数
- 磁盘
  RPM，DTR，控制器数量
- 内存
  大小
- 网络
  带宽，延迟

Servers	Server type	Configuration	RAM	Required concurrent users	Max capacity (users)	Map Images per second
GDB Minimum	Unix IBM P690	IBM p690 4 core (2 chip) 1900 MHz	8 GB	40-60	248	28
GDB Maximum	Unix IBM P690	IBM p690 8 core (4 chip) 1900 MHz	16 GB	40-60	495	54
App.Server Minimum	Windows HP Proliant 580	HP DL580 Intel Xeon 2 core (2 chip) 3200 MHz	4 GB	40-60	16	3
App.Server Maximum	Windows HP Proliant 580	HP DL580 Intel Xeon 4 core (2 chip) 3200 MHz	8 GB	40-60	40	5
App. Server Minimum	Windows IBM x3755 (#6)	IBM System X 3755 (AMD Opteron 8222 SE 2 core (2 chip) 3000 MHz	4 GB	40-60	54	4
App.Server Maximum	Windows IBM x3755 (#6)	IBM System X 3755 (AMD Opteron 8222 SE 4 core (2 chip) 3000 MHz	8 GB	40-60	110	8
App. Server Minimum	Windows IBM x3850	IBM System x (Intel Xeon E7210) 2 core (2 chip) 2400 MHz	4 GB	40-60	32	4
App. Server Maximum	Windows IBM x3850	IBM System x (Intel Xeon E7210) 4 core (2 chip) 2400 MHz	8 GB	40-60	65	8

软件选择策略

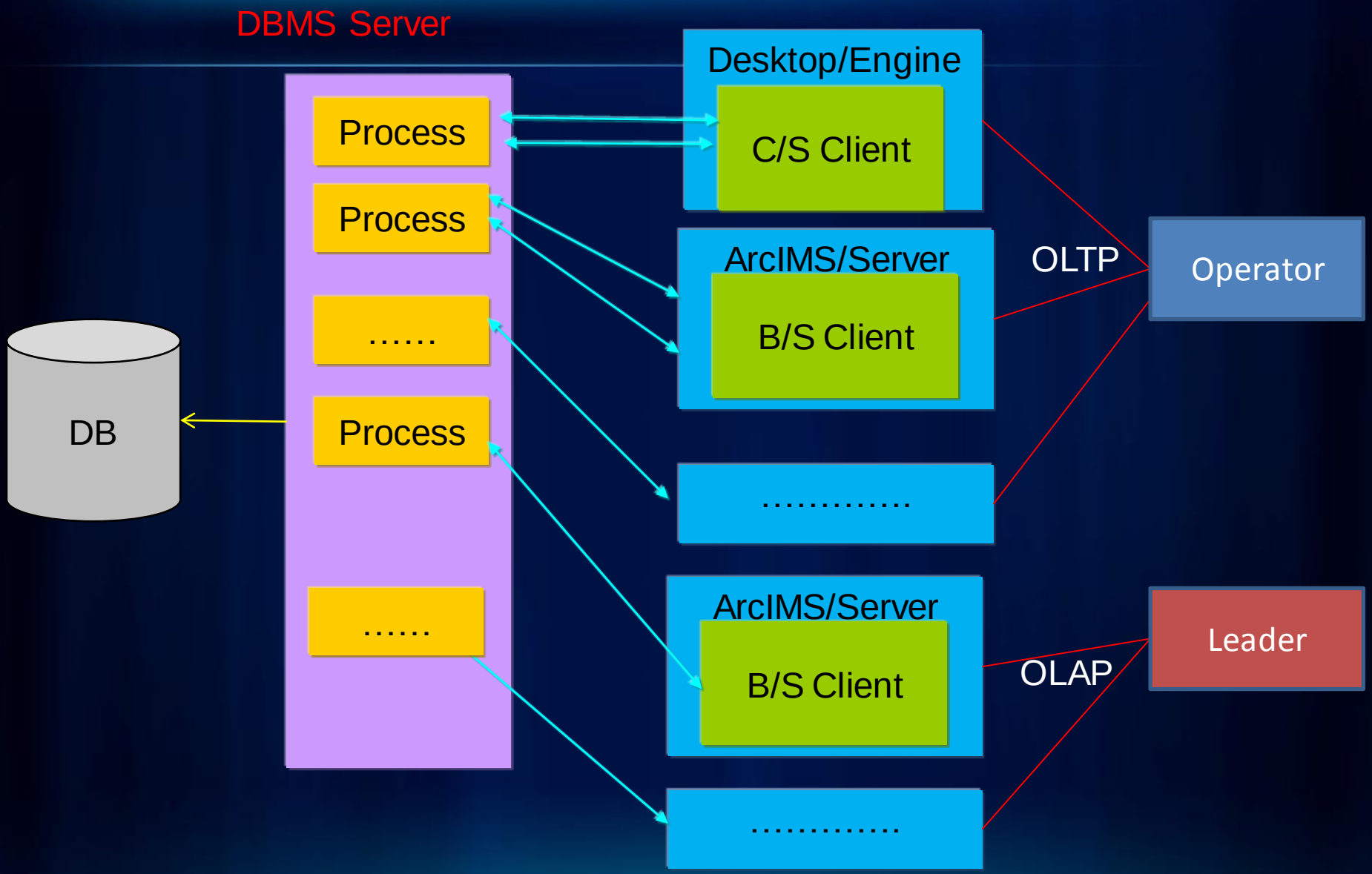
- 数据库服务器
  - OS
  - DBMS
  - ArcSDE
- 应用服务器
  - OS
  - ArcGIS
  - WebServer

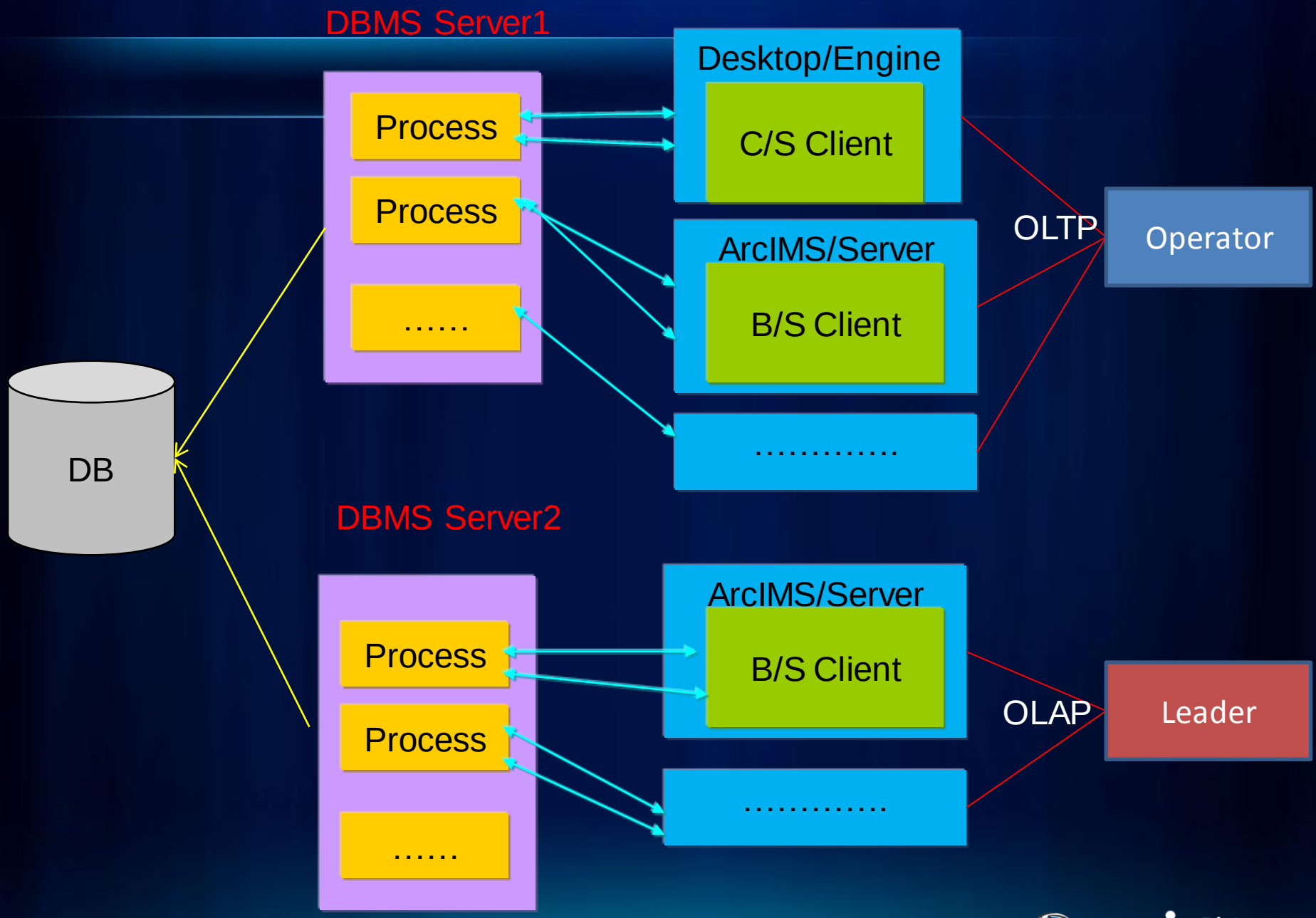
操作系统策略

- 数据库服务器
  - 计算内存和文件内存
  - 分页策略
  - 应用程序之间所占内存的比例

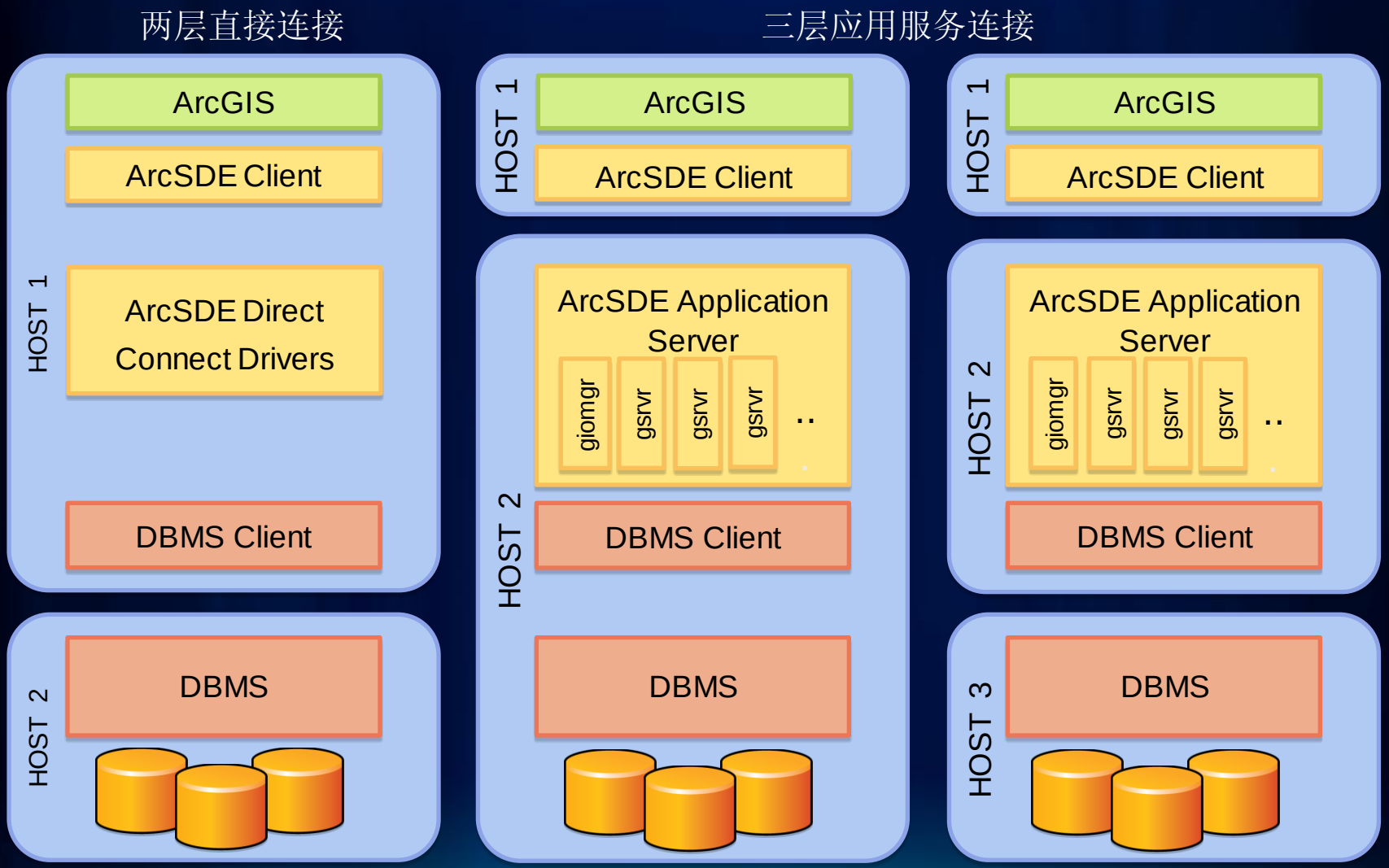
数据库优化策略

- 是否集群
- 相应的参数，存储配置
- 每个节点满足的应用类型(OLTP，OLAP)


DBMS Server
Process
Process
……
Process
……
DB
Desktop/Engine
C/S Client
ArcIMS/Server
B/S Client
……
ArcIMS/Server
B/S Client
……
OLTP
OLAP
Operator
Leader

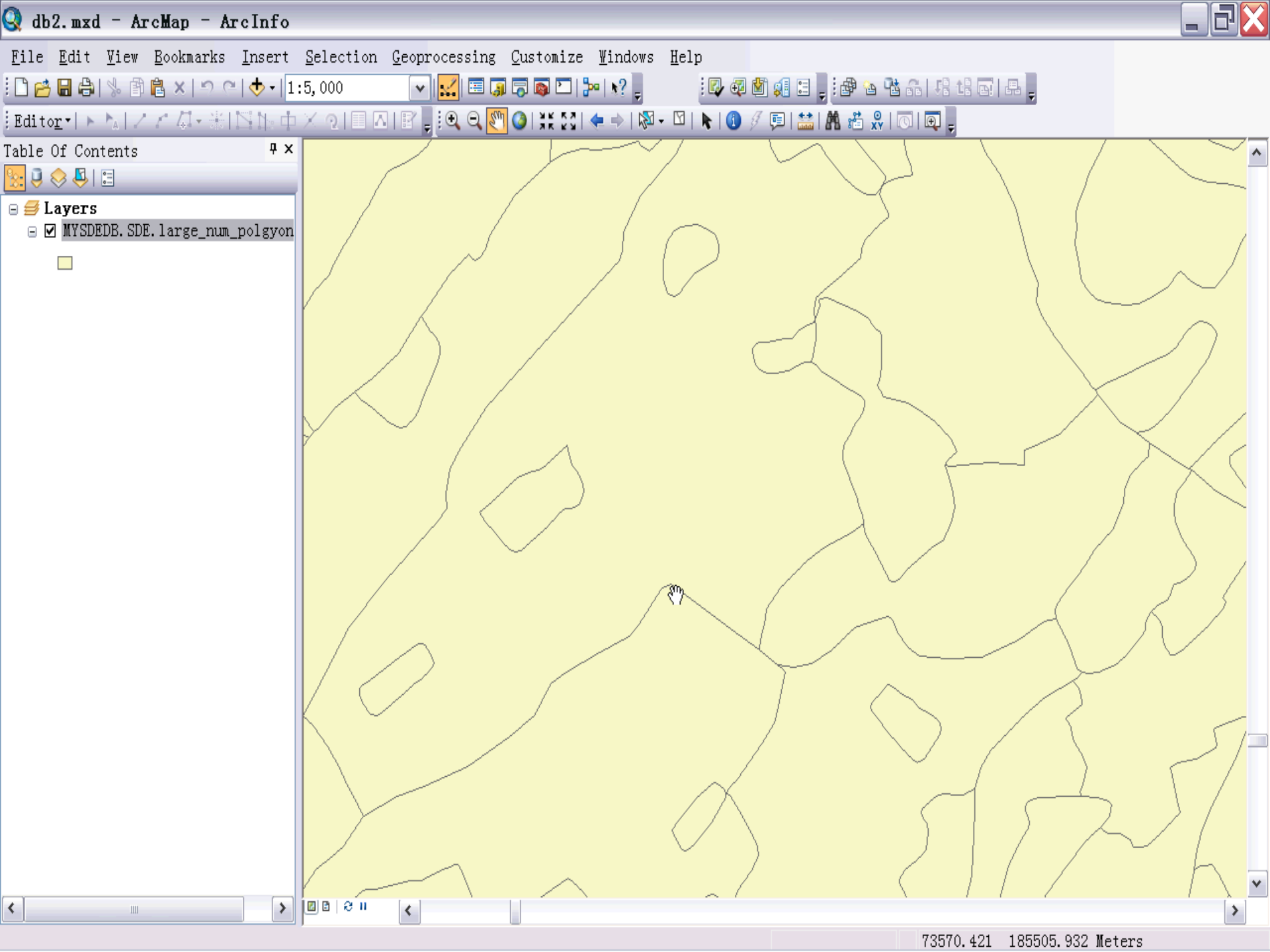


ArcSDE部署策略


两层直接连接
三层应用服务连接
HOST 1
ArcGIS
ArcSDE Client
ArcSDE Direct Connect Drivers
DBMS Client
HOST 2
DBMS
HOST 1
ArcGIS
ArcSDE Client
HOST 2
ArcSDE Application Server
giomgr
gsrvr
gsrvr
gsrvr
..
DBMS Client
DBMS
HOST 1
ArcGIS
ArcSDE Client
HOST 2
ArcSDE Application Server
giomgr
gsrvr
gsrvr
gsrvr
..
DBMS Client
HOST 3
DBMS

数据组织策略

- 组织组织模式
  - 按小单位组织
  - 按中层单位组织
  - 一张图


db2.mxd - ArcMap - ArcInfo
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:5,000
Editor
Table Of Contents
Layers
MYSDEDB.SDE.large_num_polgyon
73570.421 185505.932 Meters

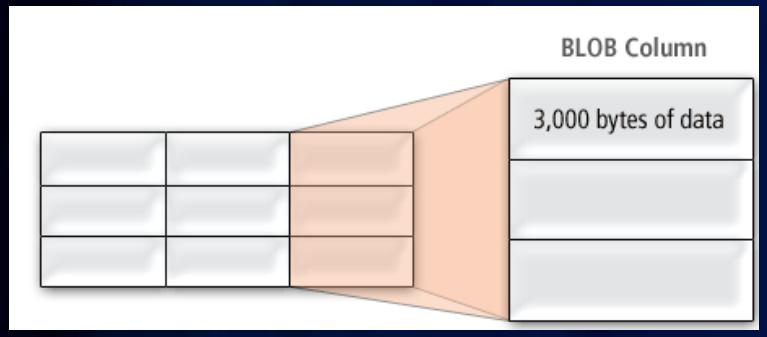
矢量数据存储类型策略

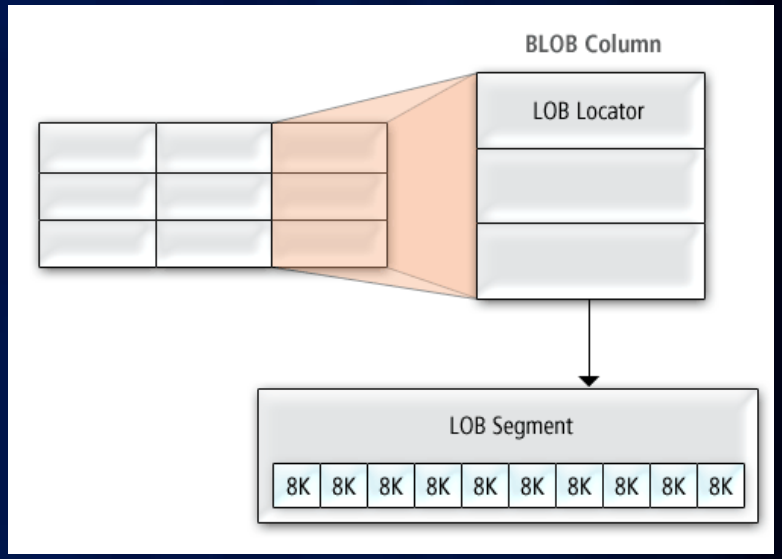
- Oracle数据库
  - ST_GEOMETRY
  - SDELOB
  - SDO_GEOMETRY

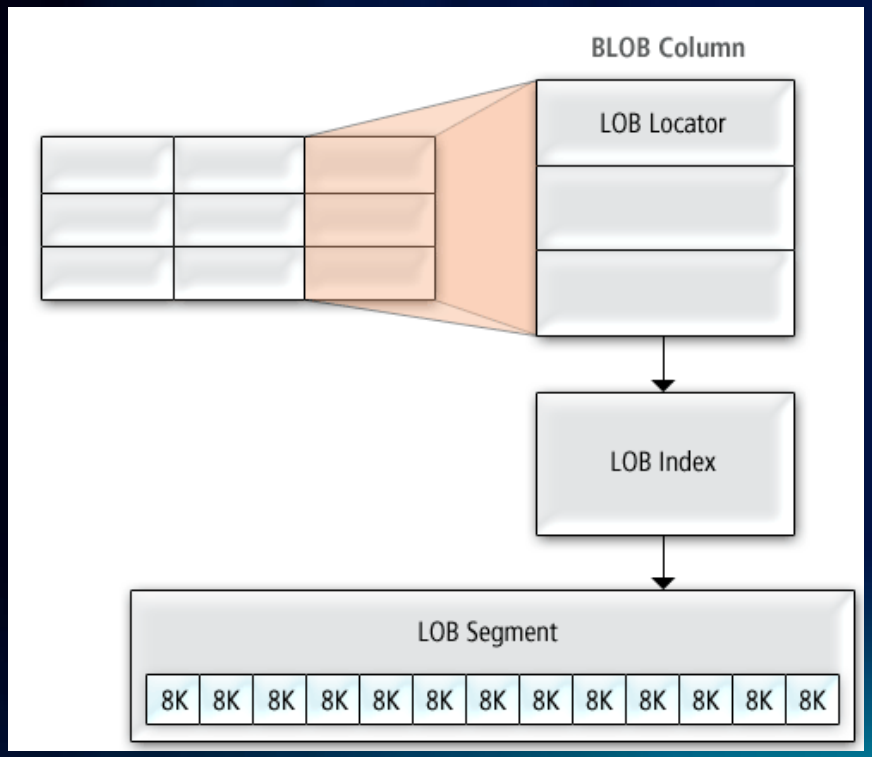
存储对比

- **PARCEL**: 783159 features in business table
  - SDELOB: 438 Mb
    - Business and F table, and S_IDX1 index
  - ST_GEOMETRY: 412 Mb
    - Business table and S_IDX1 index
    - No F-table created
  - SDO_GEOMETRY: 584 Mb
    - Business and MDRT table
    - No F-table created

BLOB相关策略







3192/16=199.5

BLOB存储参数设置

- ENABLE STORAGE IN ROW(ON)
- CACHE(ON)
- PCTVERSION(business)
- CHUNK(?)

```
ST_GEOM_LOB_STORAGE       " STORE AS ( ENABLE STORAGE IN ROW
CHUNK 8K PCTVERSION CACHE  ) "
```

确定CHUNK大小

- 取一样本数据(200-300W)
- 将样本数据录入到数据库中
- 利用DBMS_LOB包计算样本数据LOB段的平均大小

```
SQL> select trunc(avg(dbms_lob.getlength(p.shape.points)))||' Bytes' BLOB_SIZE,
trunc(avg(dbms_lob.getlength(p.shape.points)))/16||' 个' Points from catchment p;

BLOB_SIZE  POINTS
---------- ----------
2086 Bytes 130.375 个
```

B表，S表，Index表空间分离

DBTUNE

- DB2

UI_TEXT	“User Interface text for DEFALUTS”
B_RUNSTATS	"YES"
B_STORAGE	"in regtbs INDEX in idxtbs LONG IN lobtbs"

- Oracle:

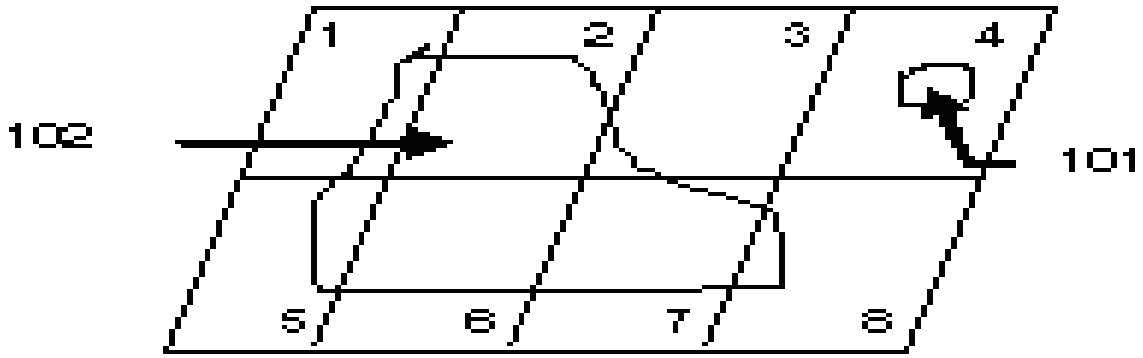
UI_TEXT	"User Interface text for DEFAULTS"
S_STORAGE	"PCTFREE 0 INITRANS 4 TABLESPACE S_TBS"
GEOMETRY_STORAGE	"ST_GEOMETRY “
B_STORAGE	"PCTFREE 0 INITRANS 4 TABLESPACE B_TBS “
B_INDEX_ROWID	"PCTFREE 0 INITRANS 4 NOLOGGING INDEX_TBS “

索引策略

- 属性索引
- 空间索引

- GRID(DB2,Oracle—ST_GEOMETRY,SQLSERVER)
- R-TREE(Informix,PostGSQL,OracleSDO_GEOMETRY)

Grid索引



空间索引(GRID)调整策略

- DB2

  gseidx
- Oracle

  sdelayer -o si_stats

```
C:\Documents and Settings\liufeng>gseidx connect to mysdedb user sde using sde g
et geometry statistics for column sde.catchment(shape) using grid sizes(6800)


行数: 2035
非空几何图形数: 2035
空的几何图形数: 0
空值数: 0

数据涉及的范围:
    X 的最小值: -355800.523149
    X 的最大值: -76469.016712
    Y 的最小值: 594210.138692
    Y 的最大值: 804249.132295



网格级别 1
------------

网格大小                                : 6800
几何图形数                              : 2035
索引条目数                              : 4985

占用的网格单元格数                      : 421
“索引条目/几何图形”比率                : 2.449631
“几何图形/网格单元格”比率              : 4.833729
每个网格单元格的最大几何图形数          : 23
每个网格单元格的最小几何图形数          : 1

索引条目数    :  1      2      3      4      10
--------------  ------ ------ ------ ------ ------
绝对值        :  494    861    12     639    29
百分比 (%)    :  24.28  42.31  0.59   31.40  1.43
```

索引调整策略

- 一级索引足够，除非你的Feature面积变化非常大
- 索引的记录数/feature的记录数<2
- 每个网格中的Feature数量在100-300为最好，尽量不要超过4000
- 一个Feature尽量不要跨越多个网格

CACHE策略1

- PACKAGE
- PACKAGE BODY
- TYPE
- TYPE BODY
- SEQUENCE(高并发编辑)

1. USER_OBJECTS
2. EXEC DBMS_SHARED_POOL.KEEP(:NAME,:FLAG)

@ ?/rdbms/admin/dbmspool

CACHE策略2

- 索引表缓存

ST_GEOMETRY--→ S<ID>_IDX$

SDO_GEOMETRY--→MDRT_ID$

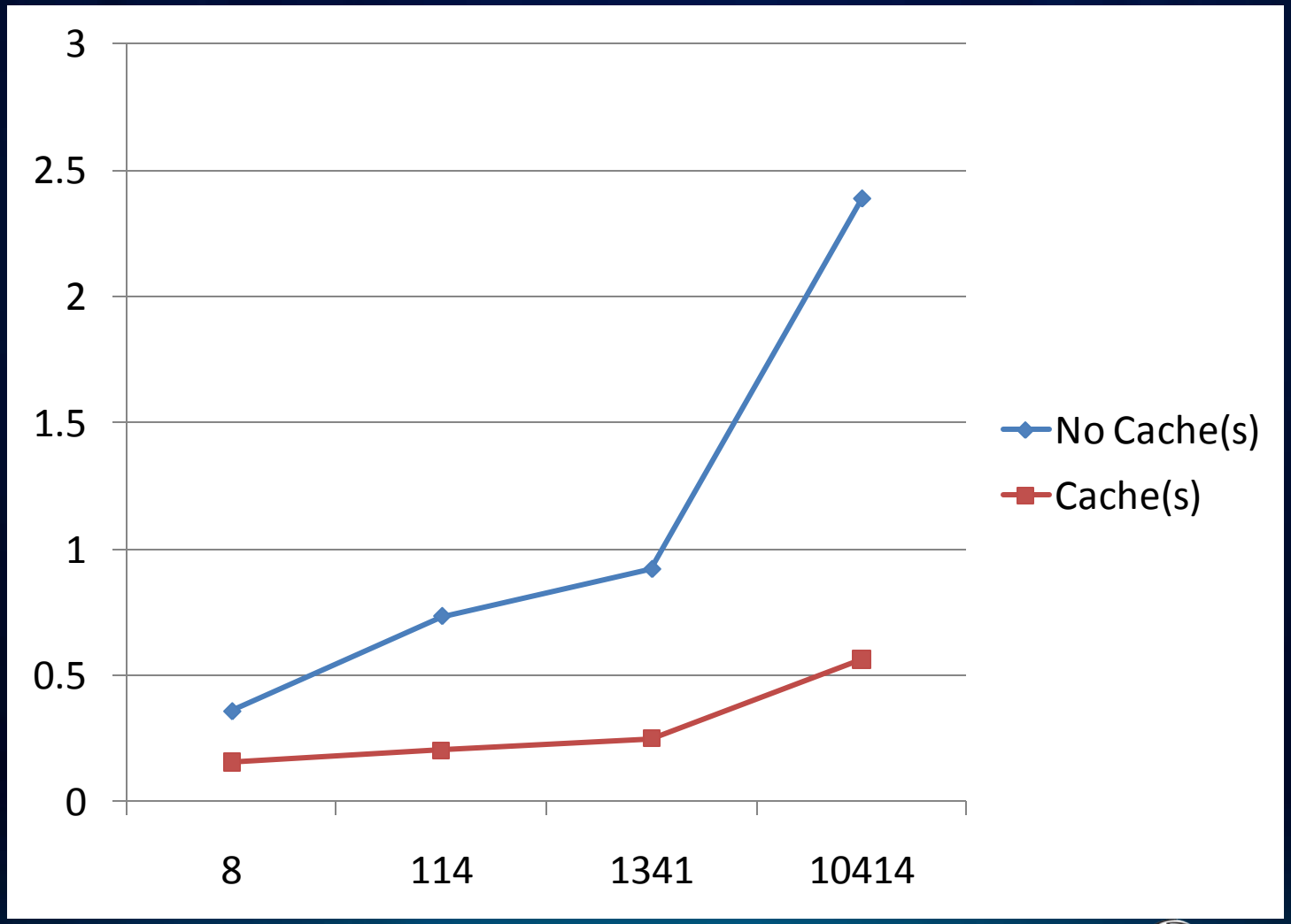
USER_INDEX_GEOM_META

ALTER TABLE TABLE_NAME CACHE.

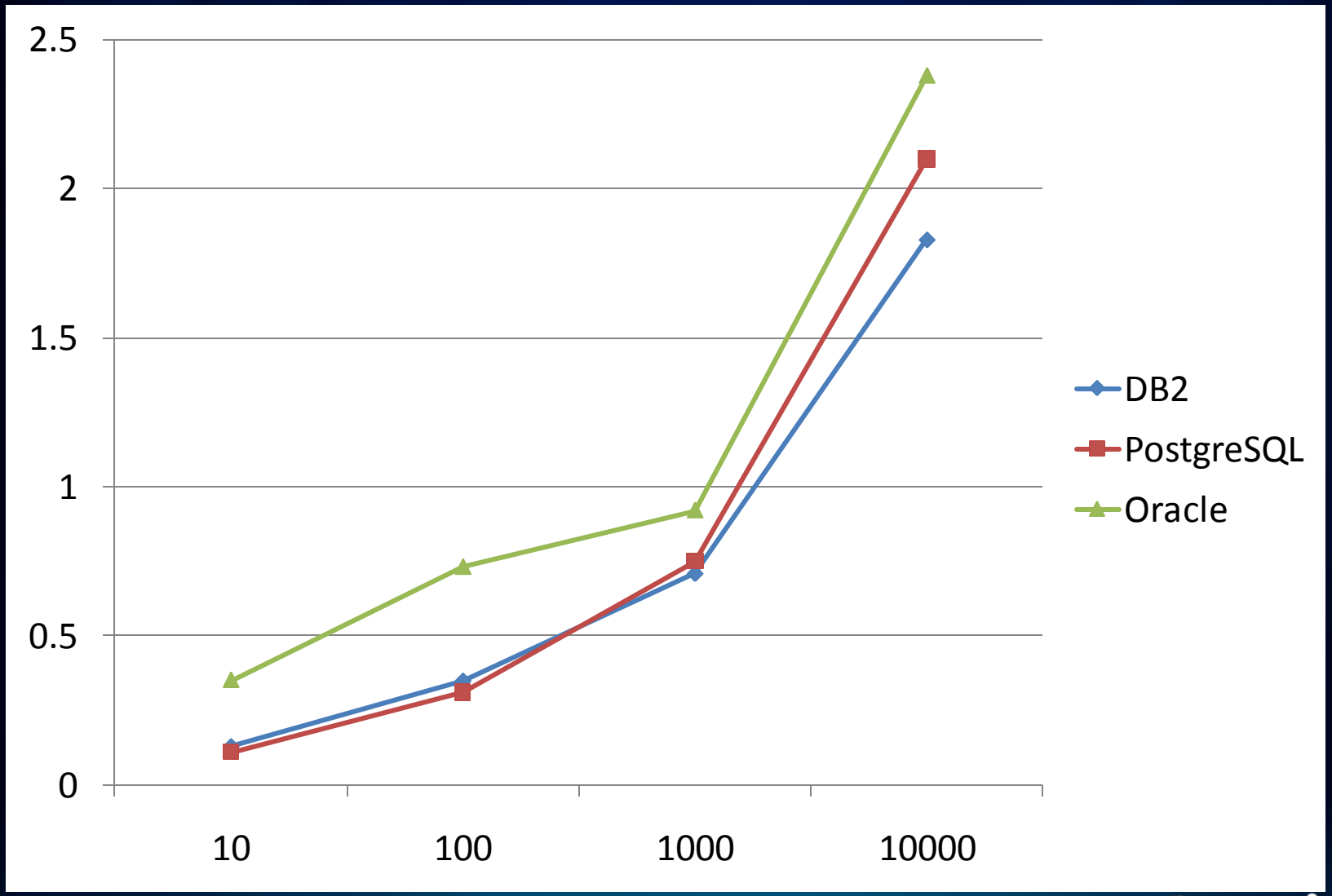
存储大小(ORACLE)

图层，表类型，表名	存储类型	记录数量	存储大小(G)	合计大小(G)
Polygon,Business Table,Polygon	SDE.ST_GEOMETRY	100880302	49.93	55.20
Polygon,Spatail IndexTable,S8_IDX$		192237821	5.27	

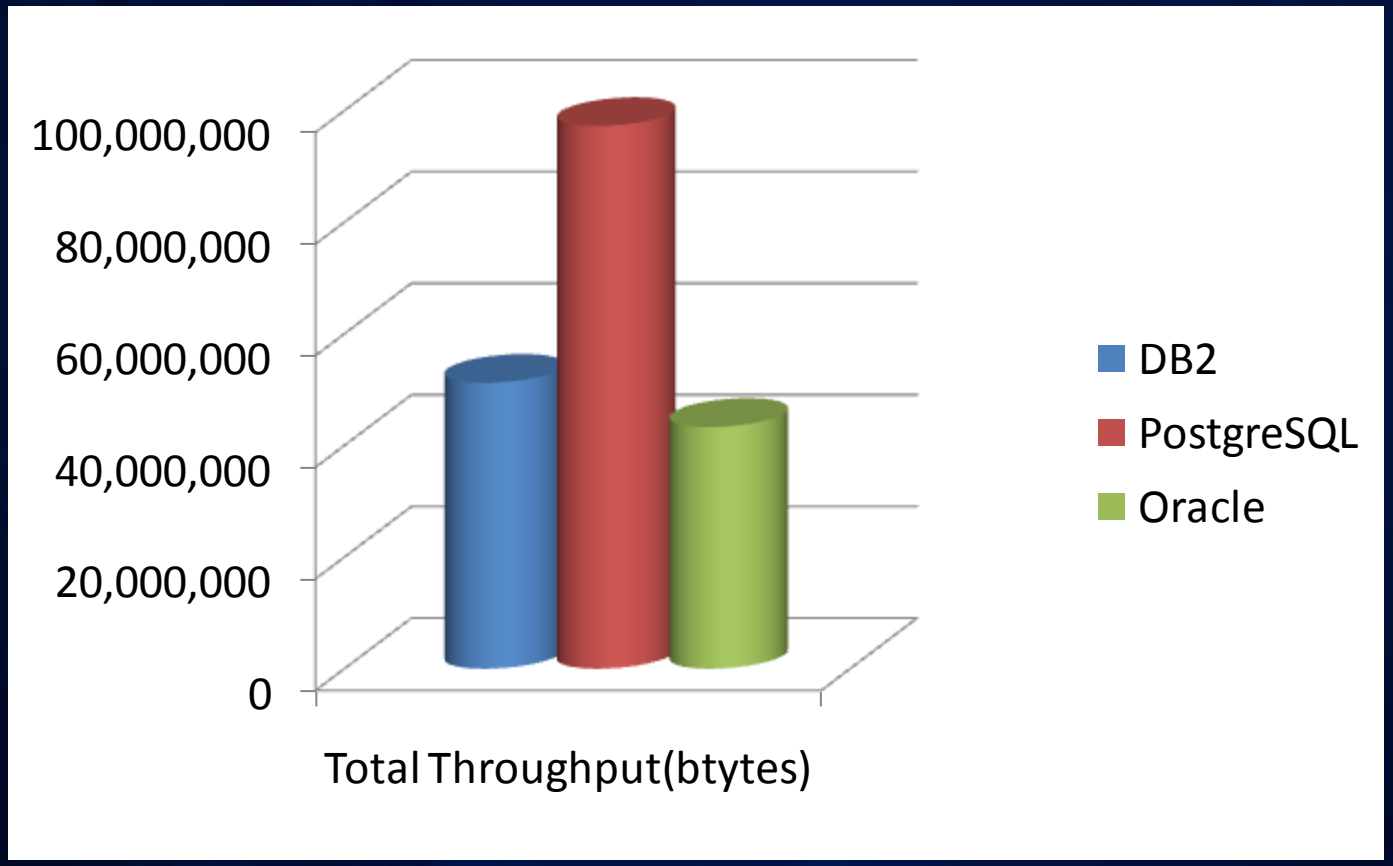
CACHE与非CACHE结果比较

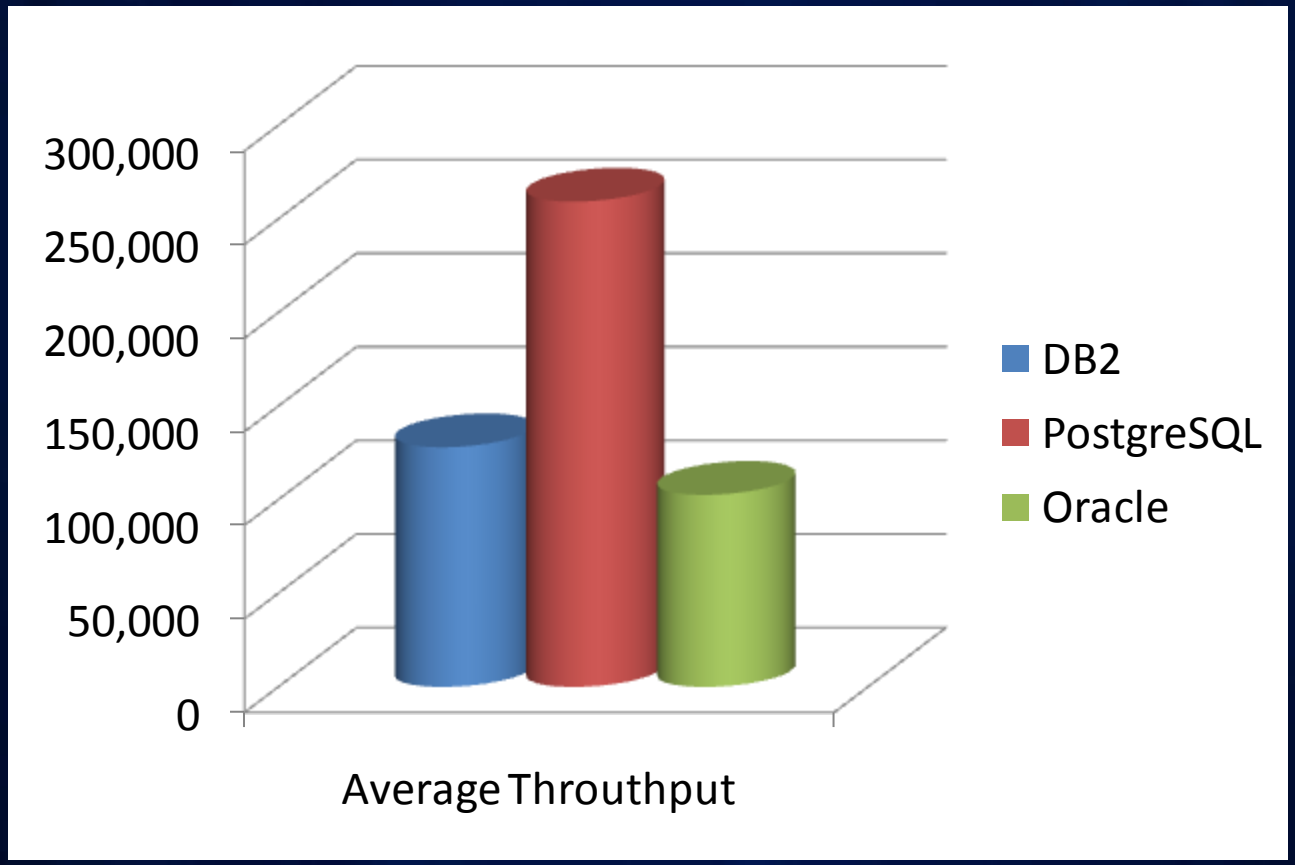


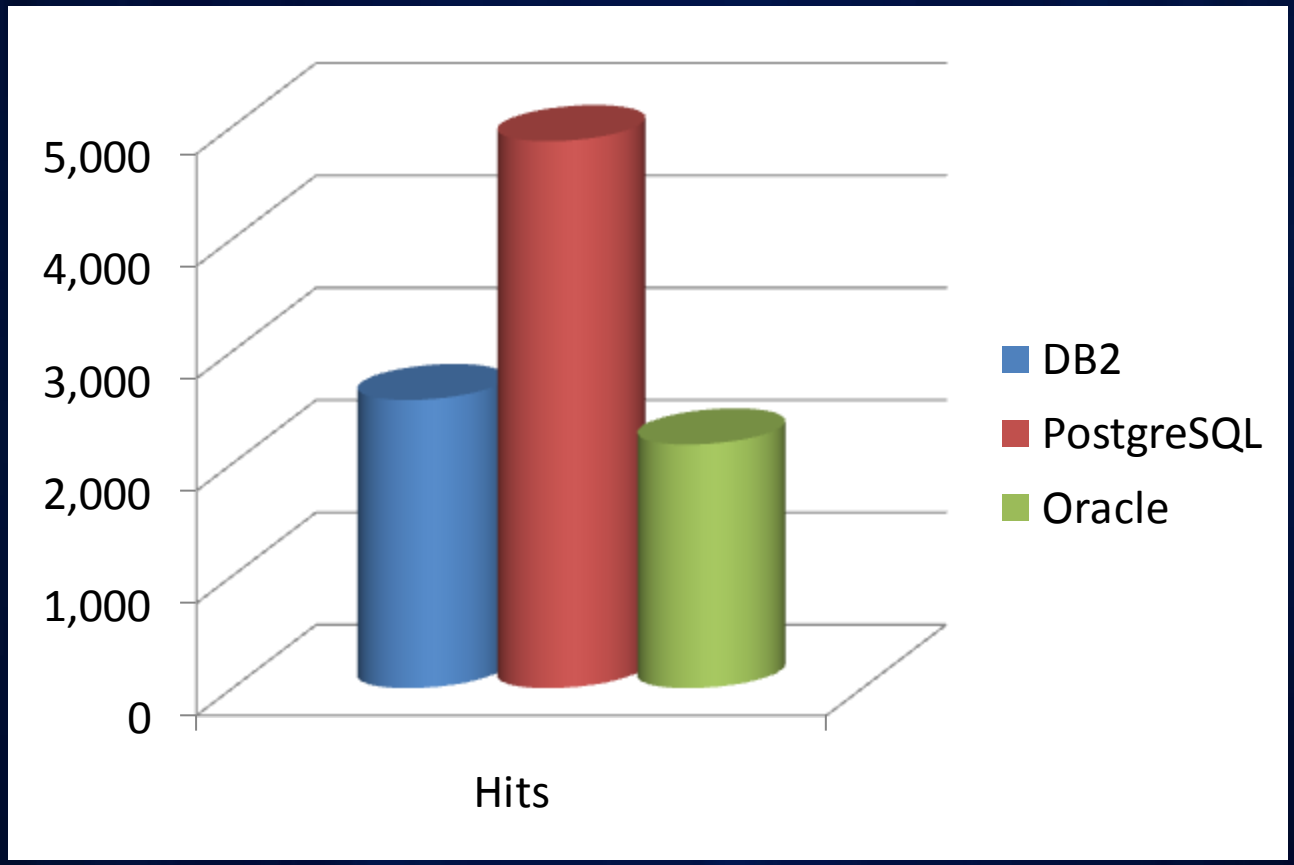
单用户大比例尺浏览



多用户大比例尺浏览


100,000,000
80,000,000
60,000,000
40,000,000
20,000,000
0
Total Throughput(btytes)
DB2
PostgreSQL
Oracle


300,000
250,000
200,000
150,000
100,000
50,000
0
DB2
PostgreSQL
Oracle
Average Throuthput


5,000
4,000
3,000
2,000
1,000
0
Hits
DB2
PostgreSQL
Oracle

内容

- 序言
- 空间数据库设计
- 矢量数据实施策略
- 栅格数据的实施策略

栅格数据模型

Geodatabase中组织栅格的方式

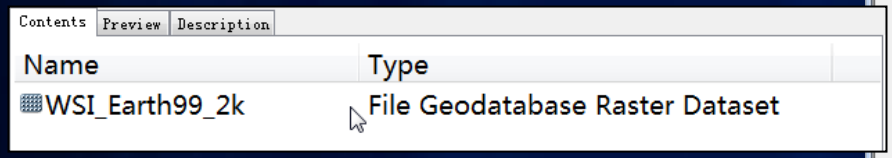
- Raster Dataset
- Raster Catalog
- Mosaic Dataset
- Feature的栅格字段

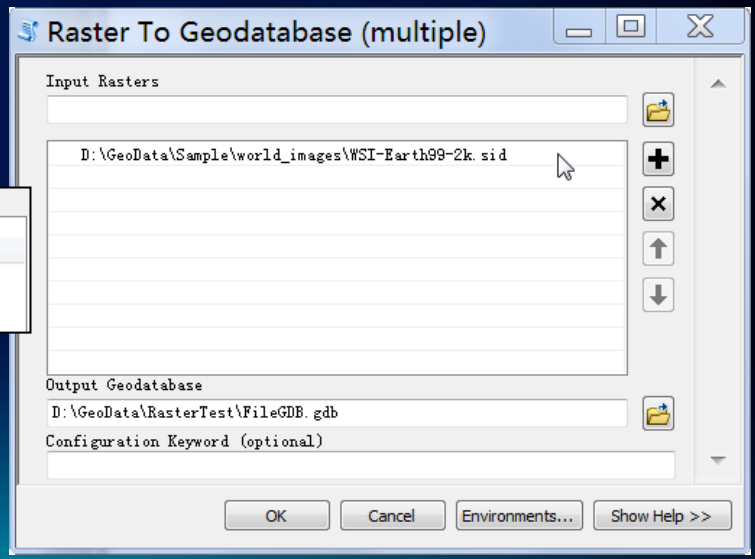
ArcGIS 10

文件格式

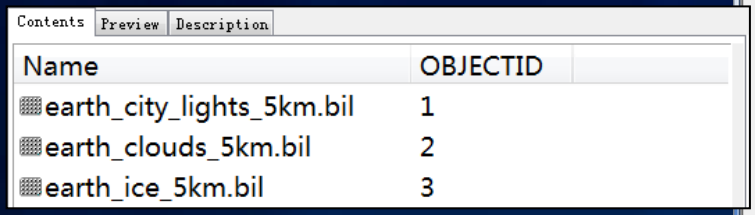
Raster Dataset与Raster Catalog

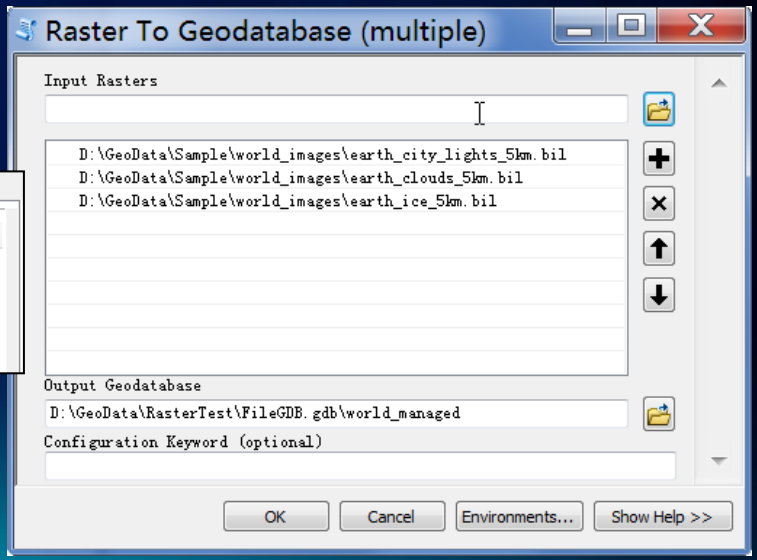
- Raster Dataset可以被认为是单幅栅格数据。
- 单张影像导入
- 多张影像导入Mosaic





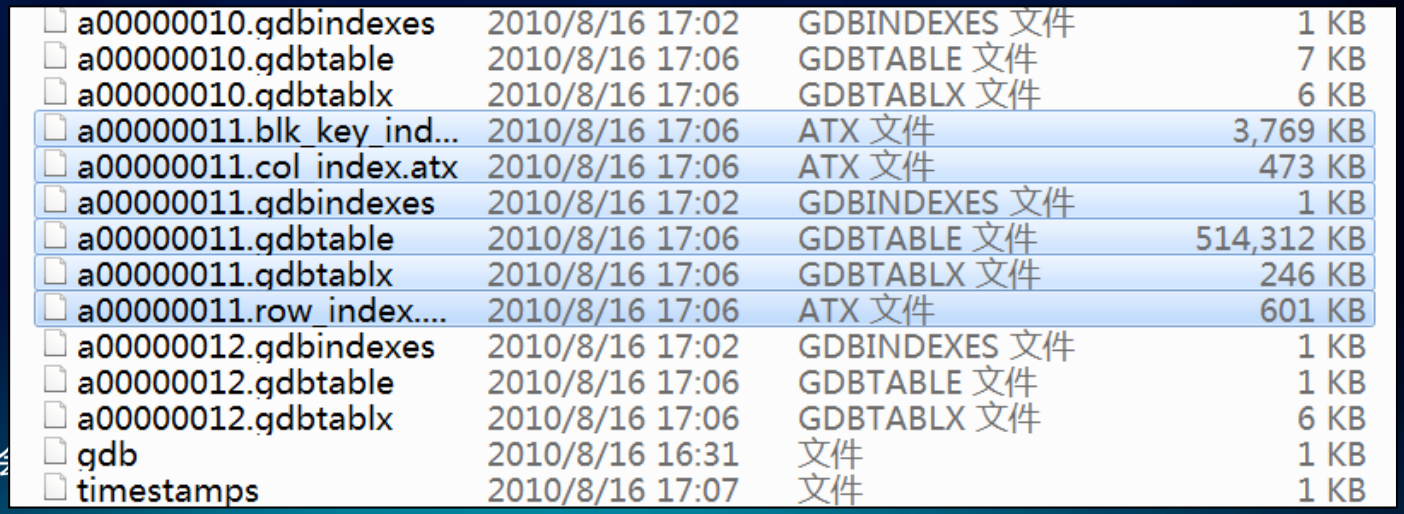
- Raster Catalog可以被认为是Raster Dataset的集合





FileGDB中的Raster Dataset

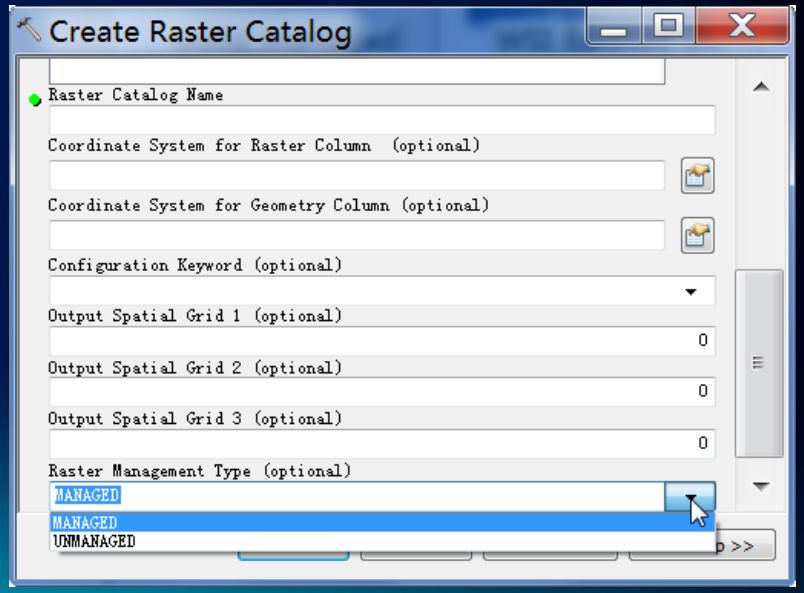
- FileGDB中Raster Dataset由若干个文件组成的
- 主要在gdbtable文件中保存所有的栅格像素值



a00000010.gdbindexes	2010/8/16 17:02	GDBINDEXES 文件	1 KB
a00000010.gdbtable	2010/8/16 17:06	GDBTABLE 文件	7 KB
a00000010.gdbtablx	2010/8/16 17:06	GDBTABLX 文件	6 KB
a00000011.blk_key_ind...	2010/8/16 17:06	ATX 文件	3,769 KB
a00000011.col_index.atx	2010/8/16 17:06	ATX 文件	473 KB
a00000011.gdbindexes	2010/8/16 17:02	GDBINDEXES 文件	1 KB
a00000011.gdbtable	2010/8/16 17:06	GDBTABLE 文件	514,312 KB
a00000011.gdbtablx	2010/8/16 17:06	GDBTABLX 文件	246 KB
a00000011.row_index....	2010/8/16 17:06	ATX 文件	601 KB
a00000012.gdbindexes	2010/8/16 17:02	GDBINDEXES 文件	1 KB
a00000012.gdbtable	2010/8/16 17:06	GDBTABLE 文件	1 KB
a00000012.gdbtablx	2010/8/16 17:06	GDBTABLX 文件	6 KB
gdb	2010/8/16 16:31	文件	1 KB
timestamps	2010/8/16 17:07	文件	1 KB

FileGDB中的Raster Catalog

- 在File Geodatabase中创建Raster Catalog时有一个很重要的选项：管理类型
- MANAGED和UNMANAGED



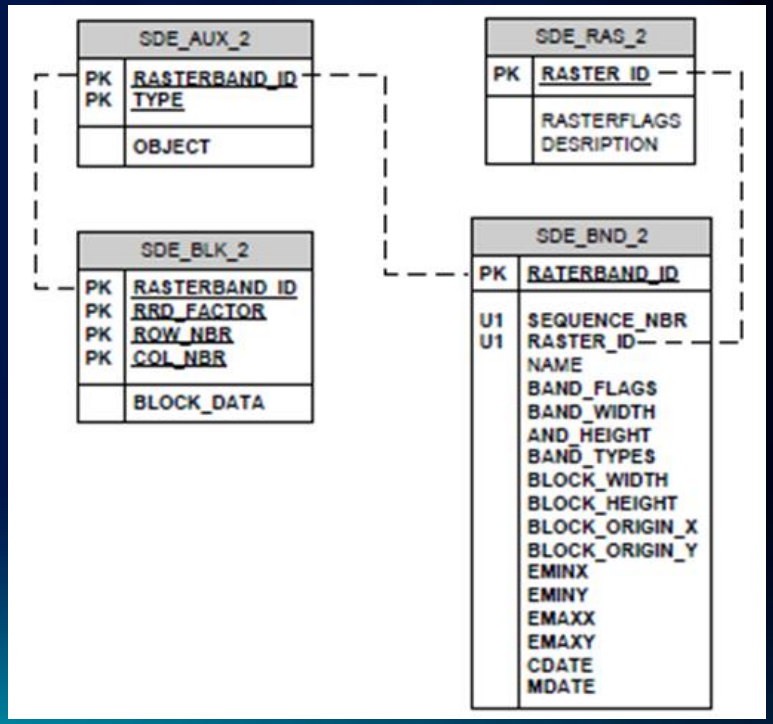
- MANAGED（托管）意味着所有的数据都将被“物理”地导入到Geodatabase中
- UNMANAGED（非托管）则对应将数据保存在文件中，而在Geodatabase中只存储指向这个真实数据的信息。
- 如果选择托管方式，那么最后在File Geodatabase中包含了若干组文件存储了这些Raster Catalog中的栅格数据的信息
- 数据量最大的所有栅格的像素数据全部存储在其中的一个gdbtable中

- 从数据管理的角度上，非托管的模式导入数据非常快，因为它无需真正将数据存储到Geodatabase中；而托管模式意味着传统的“入库”，需要耗费相当的时间

ArcSDE中的Raster Dataset和Raster Catalog

- ArcSDE中不存在非托管的方式，所有的数据都必须物理地进行导入，但是在数据库中存在不同的存储类型。
- 以Oracle为例，栅格数据可以以LONG RAW（将被Oracle废弃）、BLOB、ST_Raster或SDO_GeoRaster等类型进行存储。
- 下面看看BLOB类型存储的Raster Dataset和Raster Catalog在数据库中是如何组织的。

- 以BLOB存储的Raster Dataset/Catalog在数据库中存储在若干张相互关联的表中，其中真正存储栅格数据的表包括：Auxiliary表、Block表、Band表和Raster Attribute表。

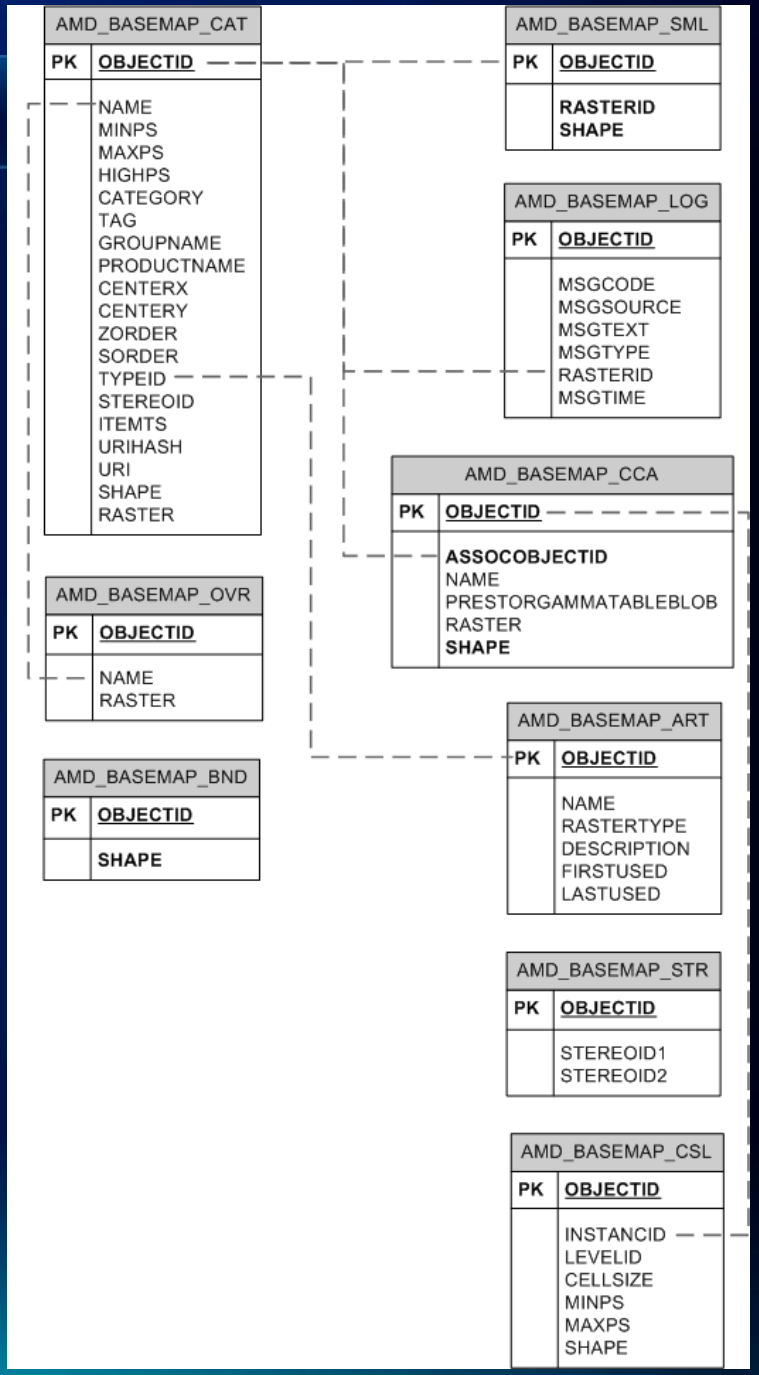


- SDE_AUX_<N>为Auxiliary表，它存储了颜色映射、图像统计信息等信息
- SDE_RAS_<N>为Raster Attribute表，它存储了Raster Dataset/Catalog中包含的“Raster”的属性，表中应该包含一条（Raster Dataset）或多条（Raster Catalog）记录
- SDE_BND_<N>为Band表，它存储了所有“Raster”的所有波段信息
- SDE_BLK_<N>为Block表、它是真正存储栅格像素值的表，Raster Dataset/Catalog中所有的栅格数据（包括金字塔）全部都以切片的形式存储在每一条记录中，一般这是数据最多的表。
- 其中<N>为RASTER_COLUMNS表中记录的RASTERCOLUMN_ID

Mosaic Dataset

- 在ArcGIS 10中出现了一种新的数据模型Mosaic Dataset。
- 在某种程度上有点类似Raster Catalog，可以管理多个栅格。
- 不管是在File Geodatabase还是在ArcSDE中，Mosaic Dataset都可以将数据保留在外部而仅在Geodatabase中保存数据的引用。
- 在ArcSDE中，Mosaic Dataset模型其实就存储在一些相关的表中：

- 总的来说，在数据库中存储的Mosaic Dataset模型的数据量是很小的。



一些影响因素

压缩格式与压缩比

- 在导入栅格数据的时候，可以根据需要选择不同的压缩格式和压缩比
- 常见的有无压缩、LZ77、JPEG、JPEG2000等
- 对于有损压缩格式还可以选择不同的压缩质量
- 对于不同的压缩格式和压缩比下的栅格数据的存储和质量，下面有一个简单的比较。

- 以一个4.72G大小的TIFF格式无压缩无金字塔的栅格数据为数据源，将其导出成若干个不同压缩格式和压缩比的数据：

压缩格式/压缩比	数据量		范围预览耗时
TIFF/无压缩	4.72G		2.2秒
PNG/LZ77	3.92G		1093.2秒
JPG/100%	2.05G	3分43秒	
JPG/75	598M	2分51秒	
JPG/50%	39		27.7秒
托管FileGDB/无压缩	4		秒
托管FileGDB/JPG/75%			0.3秒
ArcSDE/无压缩			77.6秒
ArcSDE/JPG/75%			20.3秒

大栅格数据无压缩文件存储的效率非常好

JPEG压缩算法选用75%的压缩质量是个比较好的平衡点

LZ77压缩算法压缩非常有限

JPEG压缩算法不同质量的压缩耗时相差并不太大

ArcSDE反而压缩存储比无压缩的性能要好，可见数据库存储栅格对性能影响最大的因素是读取数据的多少

File Geodatabase存储大栅格数据，即使采用JPEG压缩读取效率也不会有太大下降

栅格切片尺寸

- 在Tiled TIFF或ArcSDE中，栅格是以切片形式存储的。
- ArcSDE中存储的SDE_BLK_<N>表中，每条记录代表了一个切片。
- 默认这个切片的尺寸大小是128×128像素。

- 如果我们在导入栅格的时候选择的压缩方式为None或者默认的LZ77，数据（基本）没有被压缩。
- 128×128像素大小的一个栅格切片应该包含16K个像素。
- 对于最常见的8 Bit深度的栅格，每个像素占据1个字节；因此，这个切片将在数据库中占据16 KB存储空间。

- Oracle，默认创建数据库的数据块大小为8K。
- 上面的切片占据了两个数据块，Oracle如果要读取这个切片就需要做2个I/O操作。
- 在数据库中，I/O操作尽量需要减少，因此，所有都采用默认的设置可能并不符合实际的情况，特别是在数据量非常大的情况下。

其它存储格式

- 除了最常见的TIFF、JPEG等格式，栅格数据还可以以一些压缩格式进行存储，比如MrSID等。
- 这些格式可能有惊人的压缩比和出色的读取效率。
- 因此，在获取一些特殊的栅格存储格式的时候，最好可以比较一下它们和无压缩栅格的效率。

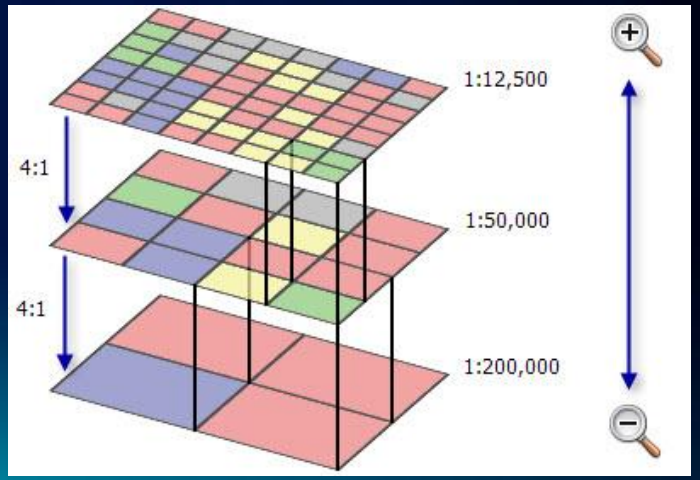
- 比如这里有一个17M的MrSID数据，将其导出为未压缩的TIFF后，两者的小范围数据预览比较如下：

压缩格式/压缩比	数据量	某小范围预览耗时
MrSID	17M	0.22秒
TIFF	622M	0.06秒

非常高的压缩比，同时数据访问的效率也不错

金字塔

- 金字塔通过在不同的比例尺下预先进行重采样并保存结果。
- 避免了原始的栅格数据在小比例尺下实时重采样的过程，因此，提高小比例尺下栅格数据的浏览效率。



- 从金字塔的原理可以知道，在与数据实际分辨率相近的比例尺下，金字塔并不能起任何加速显示的过程。
- 因此，前面的章节中分析了没有金字塔情况下的各种现象，在这种情况下完全符合。
- 同时，在创建了金字塔的比例尺下，前面的讨论也并非没有意义，它们同样也可以对金字塔本身的效率提供参考的价值。

分幅

- 一个大的栅格数据，到底是用单幅存储比较好呢，还是存为多幅较小的数据较好？换句话说，有时候拿到很多分幅的栅格数据，是不是有必要将它们都拼接起来？这里也进行一个比较。

- 比如用上面使用
据，分割成16
栅格文件，分
预览的效率比较

适度的分幅有助于提高大范围数据访问的效率。不过，如果创建了金字塔，两者的效率应该区别不大。另外，还会有压盖和色差的问题。

压缩格式/压缩比	全图预览耗时	某小范围预览耗时
单幅	26.7秒	0.08秒
多幅（16幅）	17.7秒	0.08秒

并发访问

- 对于使用文件形式存储的栅格，或许有人会对其在并发访问环境下的能力有所怀疑。在这里，分别使用文件、File Geodatabase和ArcSDE存储相同的数据，在ArcGIS Server上发布服务运行 20个实例，然后比较他们在50个用户并发访问下的性能：

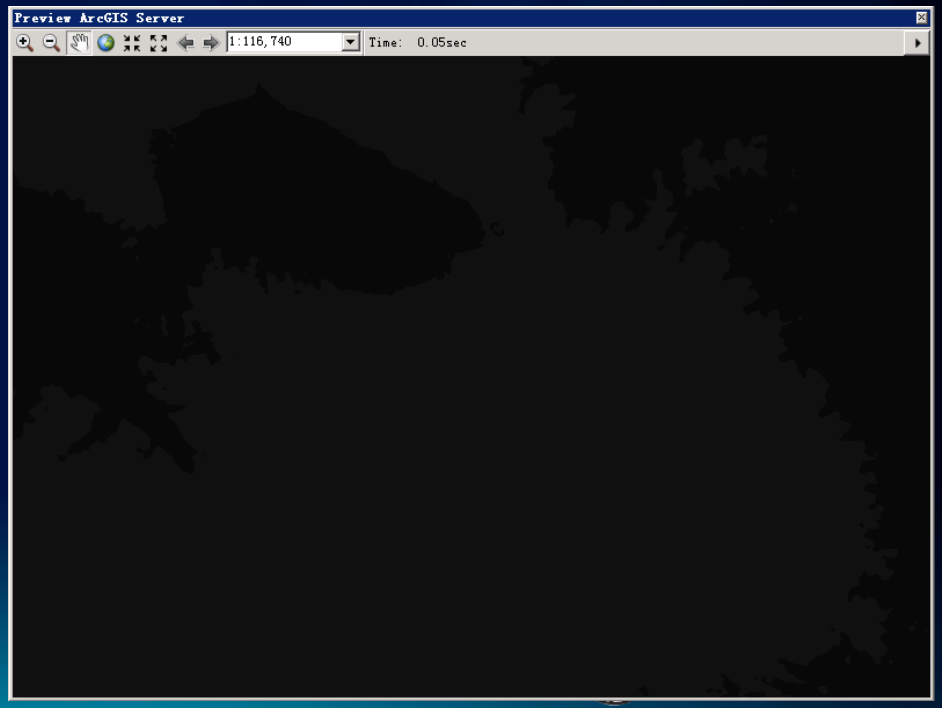
压缩格式/压缩比	单用户访问耗时	并发访问响应时间
文件（TIFF）	2.2秒	0.76秒
托管FileGDB（JPEG/75%）	20.3秒	2.13秒
ArcSDE（JPEG/75%）	20.3秒	1.78秒

设计策略示例

单幅大影像

- 数据源： 单幅58.1G大小的NTF数据。
- 这是一个无压缩的DEM数据
- 但是对于NTF这种数据格式直接访问的性能，我们未必有把握断言，因此，首先需要对这个数据本身读取的效率进行一定的度量。
- 参考前面的讨论和一些数据，我们知道一个4.72G大小的无压缩TIFF在数据分辨率下进行一个预览的耗时仅有0.08秒。现在首先来看一下这个近60G的无压缩NTF数据的效率：

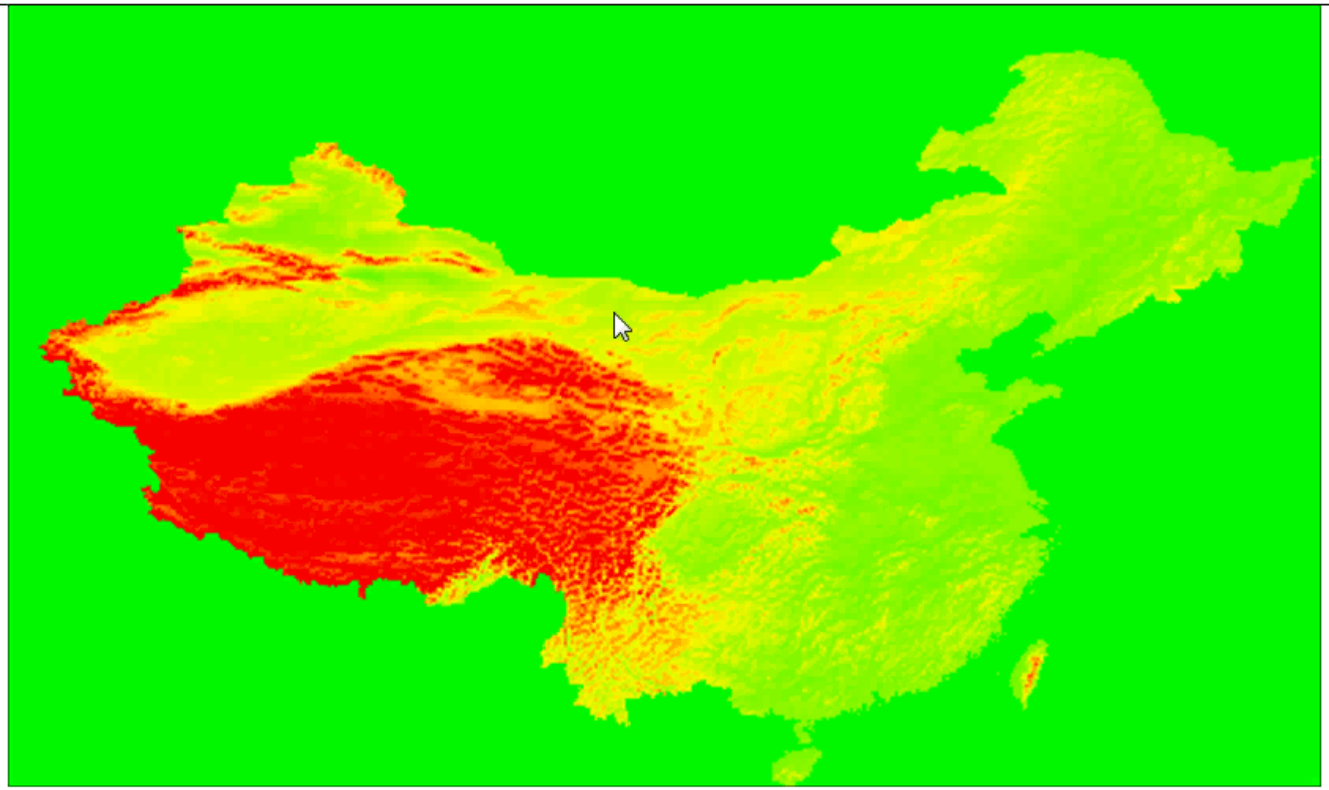
- 在栅格数据的分辨率比例尺下，通过预览工具进行渲染耗时0.05秒，可见，NTF格式无压缩数据的读取效率没有问题。



- 但是，该数据并没有金字塔，在小比例尺下的效率很低，因此，必须对其进行创建金字塔的操作。对于这个近60G的数据，创建全比例尺的金字塔耗时39分钟，生成了2G大小的金字塔。
- 直接用这个数据进行配图发布服务，使用50个并发用户进行动态出图的压力测试，可以达到5M/秒的吞吐量。

ArcGIS JavaScript API: raster/single_60g

Built using the ArcGIS JavaScript API



多幅小影像

- 数据源：146幅总量23.3G的MrSID数据。
- 这是一个本身包含金字塔的压缩数据，其未压缩大小约为750G。
- 首先面临的一个选择就是，是不是需要解压这个数据？要做这个选择的前提是需要估计解压缩这个数据需要的时间、数据以后更新的频率和存储空间的富余情况。其中最重要的可能还是解压缩的耗时。

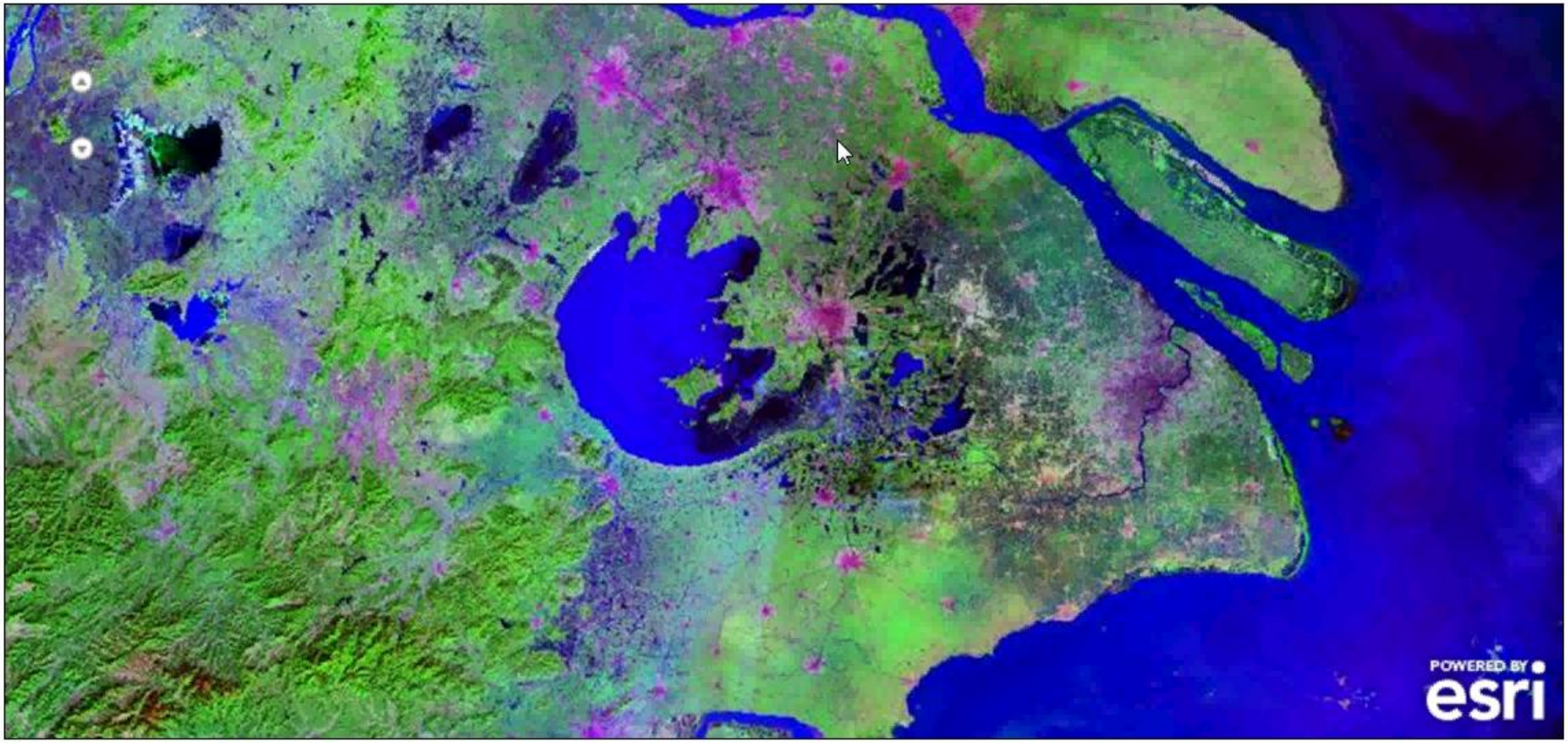
- 我们取其中一幅数据进行测试，其未压缩大小为5.2G，源数据大小为200M。测试将其解压为无压缩含金字塔的TIFF格式总共耗时1小时23分，这样保守估算所有数据如果导出为无压缩格式大概需要160小时（一周）。
- 在这里，我只想将其作为底图，因此配置为地图服务后还会进行切片缓存，完全没有必要花这么多时间在数据解压上。

- 接下来的工作就直接基于MrSID数据。
- 因为数据分幅比较多，我们希望可以用更方便地方式进行管理。但是，我们不要导出数据，当然也不希望进行托管入库操作，因为托管的过程肯定比导出无压缩格式更加耗时。这时，我们就只有两个选择，一个是File Geodatabase结合非托管模式或Mosaic Dataset、另一个是ArcSDE结合Mosaic Dataset。
- 这里我选择ArcSDE。

- 在ArcSDE中新建了一个Mosaic Dataset，然后将所有的MrSID数据直接添加到这个数据集中，耗时37分11秒，也就是说每大约15秒可以导入一个栅格的信息到这个Mosaic Dataset中。
- 在ArcGIS 10中，我们可以直接将这个Mosaic Dataset发布成Image Service，对这个服务使用50个并发用户进行动态出图的压力测试，可以达到2.5M/秒的吞吐量。

ArcGIS JavaScript API: SDE.multi_750g

Built using the ArcGIS JavaScript API



谢
谢！