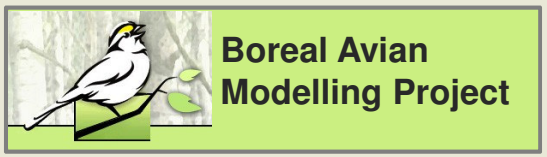
The Canadian Spatial Data Foundry

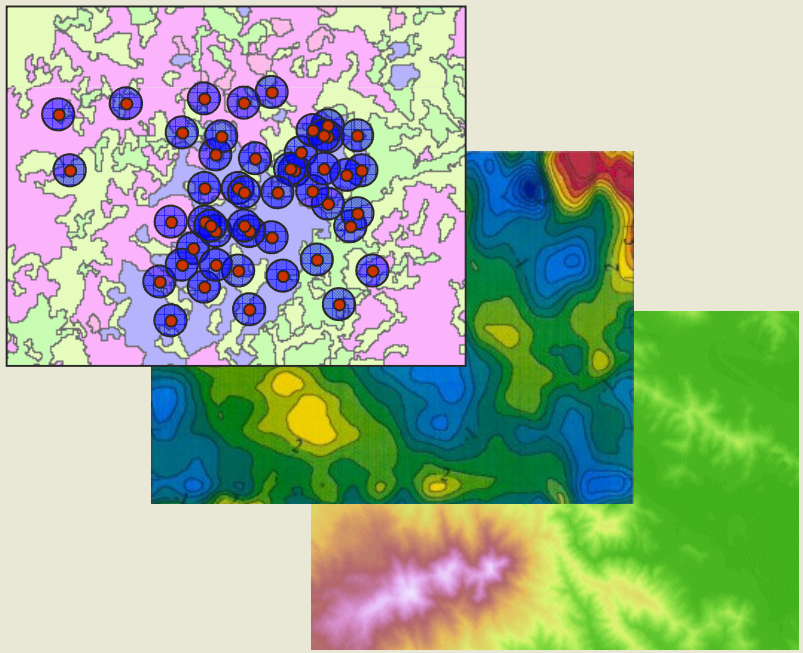
Introduction to PostGIS WKT Raster and "Raster Objects"

Pierre Racine
Professionnel de recherche
Département des sciences du bois et de la forêt



1 - The Canadian Spatial Data Foundry
The Context

- Many researchers in **forestry**, **ecology** and **environment**
- Interested in habitat selection modelling
  - Where do organisms prefer to live?



shape	obsID	cutProp	meanTemp	elevation	etc…
polygon	1	75.2	20.3	450.2	…
polygon	2	26.3	15.5	467.3	…
polygon	3	56.8	17.5	564.8	…
polygon	4	69.2	10.4	390.2	…
…		…	…	…	…

etc…

1 - The Canadian Spatial Data Foundry
The Problem

Researchers must…

…learn lots of ArcGIS, to use only a few operations

…search for, download and assemble large datasets

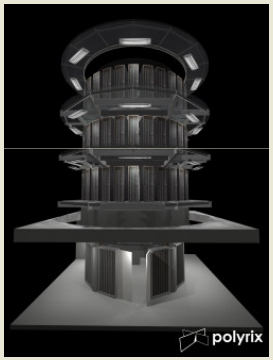
- **historical data are often lost**
- **data are delivered in many different formats**
- **datasets are too large to fit in one file (shp limited to 2 GB, complete forest cover for Canada is 30GB, complete DEM for Canada is 9 GB, etc…)**
- **computation is often too difficult for ArcGIS (800 buffers over 5 000 000 polygons)**

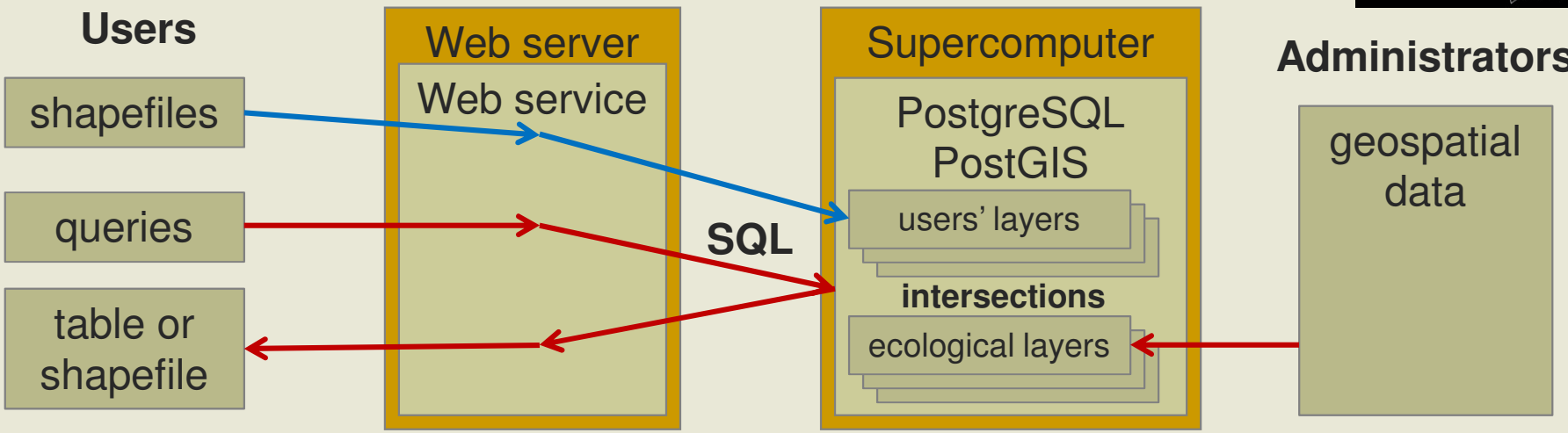
…struggle for weeks, if not months, to get their data table ready for statistical analysis…

In brief: researchers waste much energy on tasks unrelated to their main priority: research!!!

The Canadian Spatial Data Foundry
The Envisioned Solution

- Building a **paying web service**
- Backed by a **spatial database** (PostGIS) hosted on a **supercomputer**
- **Administrators** upload **preassembled** datasets of **ecological layers** (vector & raster, historical data included)
- Users with **accounts upload** their datasets (**shapefiles**)
- **Create intersection queries** on the ecological layers
- Obtain resulting **shapefiles** or **tables** (minutes, hours or days later)





The Canadian Spatial Data Foundry
What is a spatial database?

- **DBMS with native support for the geometry type**
  - Normalisation
  - Standard Query Language (SQL)
  - Transactions & Rules
  - Security & Backup
  - Functions & Operators (intersect(), within(), area(), =, &&, etc…)

shapefile

properties	
shape	owner
polygon	Jean
polygon	Pierre
polygon	Marc
polygon	Jean
…	…

spatial DB

properties	
geometry	owner
polygon	Jean
polygon	Pierre
polygon	Marc
polygon	Jean
…	…

districts	
geometry	id
polygon	AB12
polygon	CF34
polygon	RT43
polygon	RE42
…	…

```
SELECT area(geometry), owner FROM properties, districts
WHERE intersect(properties.geometry, districts.geometry) and district.id = "AB12"
```

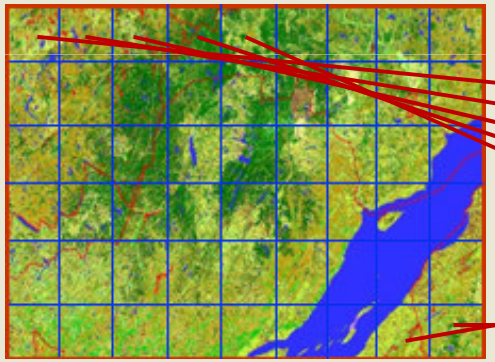
  - What is the area — and who is the owner — of properties located in district AB12?

- **IBM DB2 Spatial Extender, Informix Spatial DataBlade, Oracle Spatial, PostgreSQL/PostGIS, ESRI's ArcSDE, Intergraph's GeoMedia**
- **What about raster?**

Raster Support Requirements
1 - Storage of Non Rectangular Raster Coverage

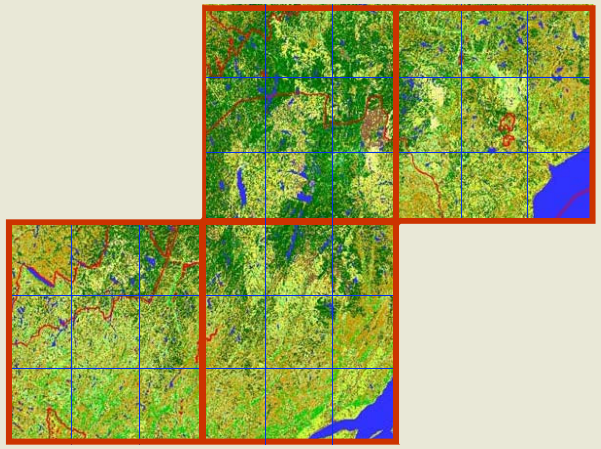
- We have to be able to store not only **"ideal" rectangular raster datasets**…

"Ideal" Raster Dataset



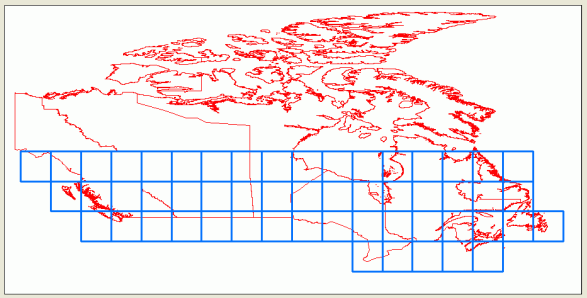
landcover	
tileId	raster
1	rasterBLOB
2	rasterBLOB
3	rasterBLOB
4	rasterBLOB
5	rasterBLOB
…	…
53	rasterBLOB
54	rasterBLOB

"Real" Raster Dataset

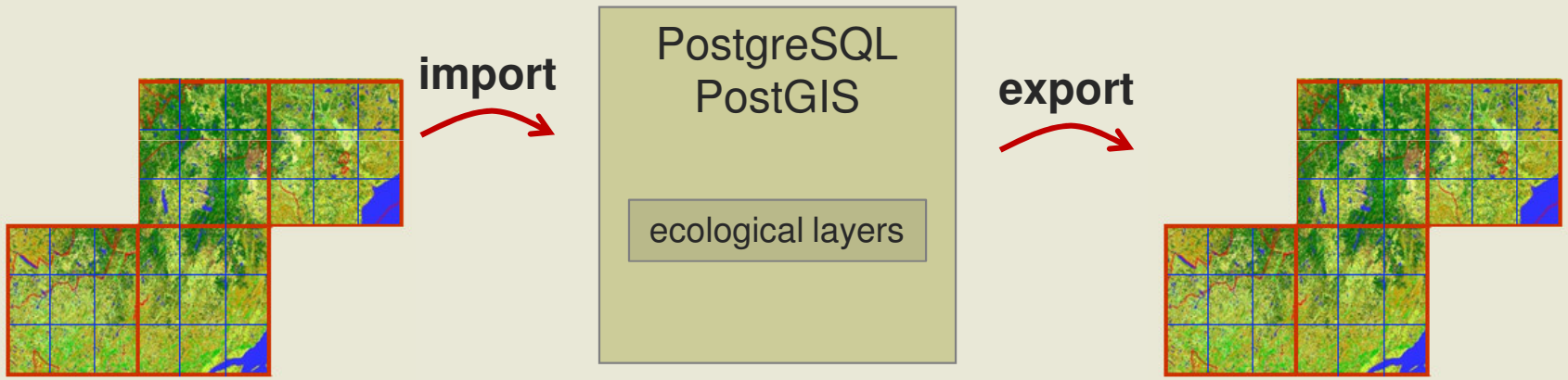


- …but also "real" **non-rectangular raster coverages**

e.g. SRTM Coverage for Canada



Raster Support Requirements
2 - Easy Importation/Exportation



The way to import raster layers should not differ much from the way to import vector layers…

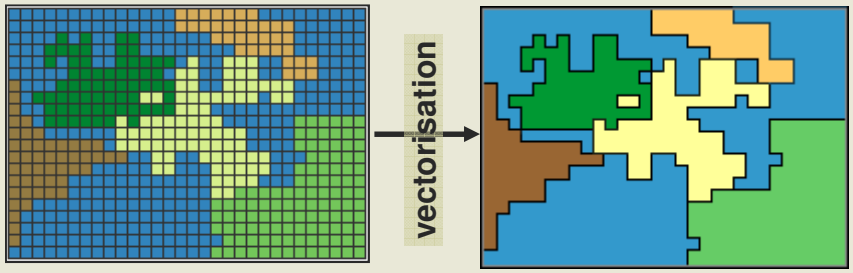
Raster Support Requirements
3 - SQL Functions & Operators on the Raster Type

- **Raster Attributes**
  - area(), srid(), width(), height(), pixeltype(), pixelsize(), nodatavalue(), georeference(), etc…
- **Raster Transformation**
  - reproject(), translate(), scale(), resample(), clip(), reclass(), mapalgebra(), etc…
- **Raster Aggregation**
  - Merge of many rasters using GROUP BY (accum())
- **Raster Conversion**
  - toJPEG(), toTIFF(), to KML(), toPolygon()…

Raster Support Requirements

4 - Lossless Conversion Between Vector and Raster Layers

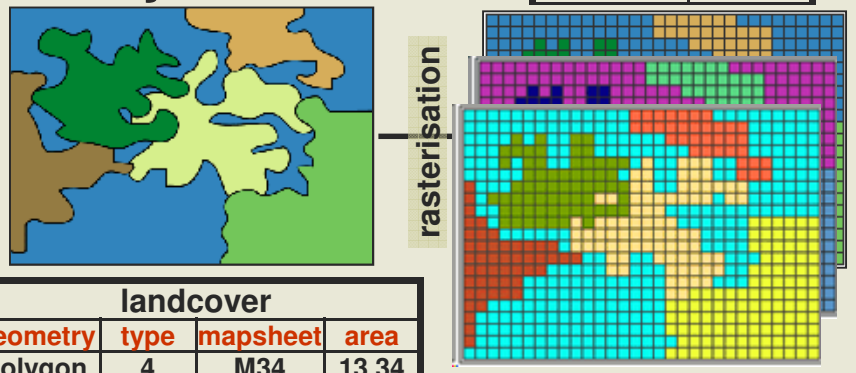
- Categorical rasters layers convert **well** to vector layers
  - one variable converts to one column
  - groups together pixels of same value
  - contiguous or not
  - continuous raster layers do not convert as well



landcover

landcover	
geometry	type
polygon	4
polygon	3
polygon	7
...	...

- Vector layers **do not** convert **well** to raster layers
  - each attribute (e.g. type) must be converted to one raster
  - no support for nominal values (e.g. "M34")
  - global values (area) lose their meaning
  - overlaps are lost
  - resolution must be high to match vector precision
  - features lose their unique identities
  - reconversion to the original vector is very difficult or impossible



area

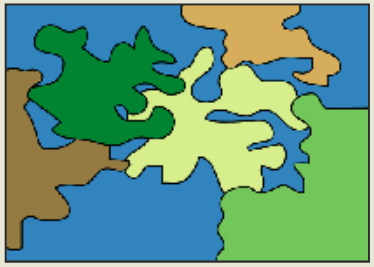
landcover			
geometry	type	mapsheet	area
polygon	4	M34	13.34
polygon	3	M33	15.43
polygon	7	M33	10.56
...	...	...	...

We need a better way to convert vector layers to rasters without destroying the objects' identities

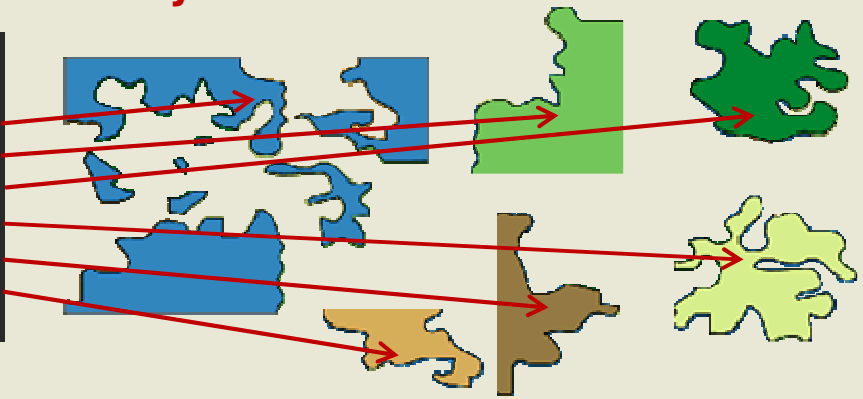
Raster Support Requirements

4 - Lossless Conversion Between Vector and Raster Layers

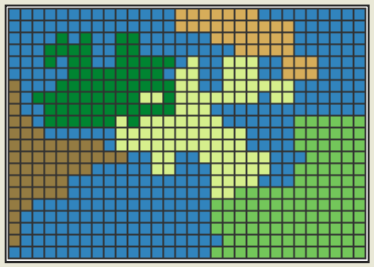
- In a **vector layer**, each object has **its own identity**



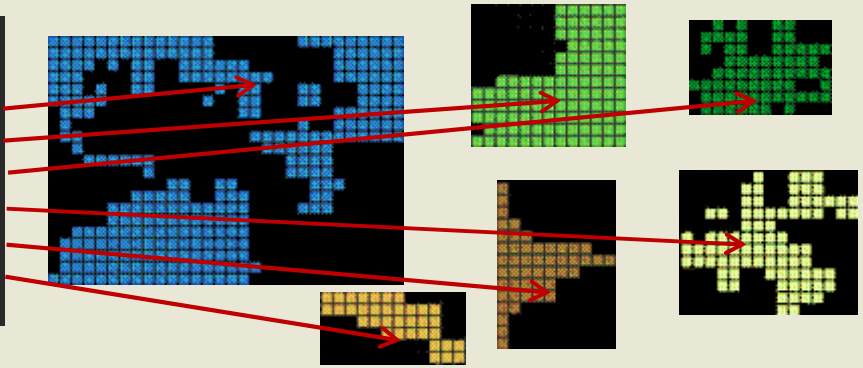
landcover			
geometry	type	mapsheet	area
polygon	4	M34	13.34
polygon	3	M33	15.43
polygon	7	M33	10.56
polygon	9	M34	24.54
polygon	5	M33	23.43
polygon	2	M32	12.34
...	...	...	...



- In a **raster layer converted** from a vector layer, each object should **conserve its own identity**



landcover			
raster	type	mapsheet	area
raster	4	M34	13.34
raster	3	M33	15.43
raster	7	M33	10.56
raster	9	M34	24.54
raster	5	M33	23.43
raster	2	M32	12.34
...	...	...	...



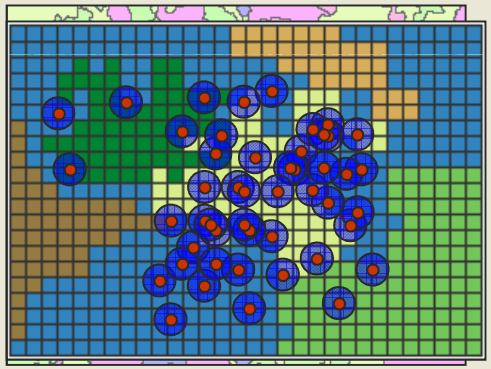
- Each "raster object" has **its own georeference**
- Black pixels are **"nodata values"**
- Like vectors, raster objects may or may not **overlap**
- **Raster algorithms** can be used on the whole layer after a **"blend"** of the objects into a single raster

Rasters become just another way to store geographic features in a more expressive vector object-oriented-like style

Raster Support Requirements

5 - Seamless Spatial Operators & Functions on Vector and Raster Types

- The goal is to be able to use **a single set of SQL functions & operators** without worrying if data are stored in vector format or raster format.
  - Same deployment strategy (SQL)
  - No longer need to implement overlay operations in two different ways



observation	
geom	obsid
polygon	24
polygon	31
polygon	45
...	...

∩

cover	
raster	ctype
raster	4
raster	3
raster	5
raster	2
...	...

=

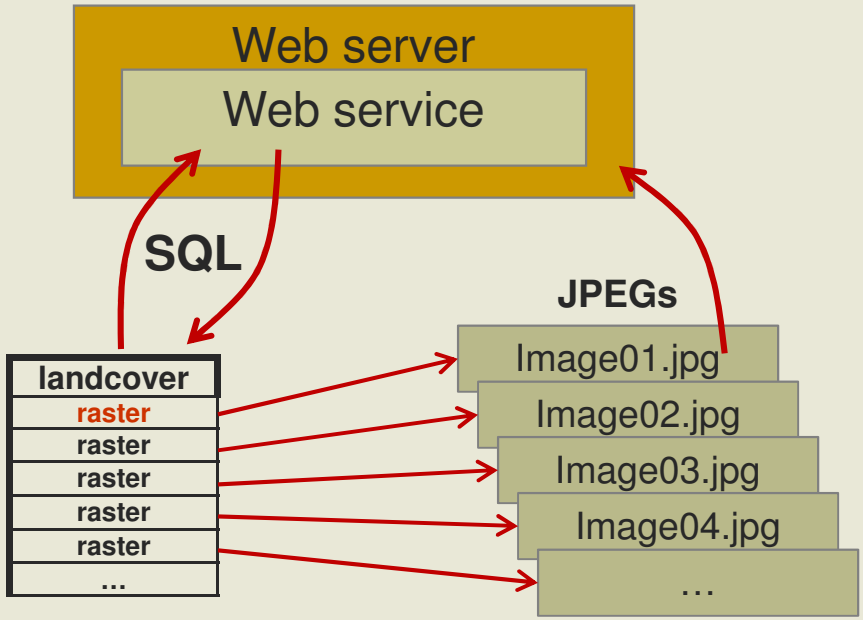
result			
geom	obsid	ctype	area
polygon	24	4	10.34
polygon	53	3	11.23
polygon	24	5	14.23
polygon	23	2	9.45
...	...	...	...

```
SELECT geom, obsid, ctype, Area(geom) as area FROM (
SELECT Intersection(Buffer(observation.geom, 1000), cover.geom) as geom, obsid, type
FROM observation, cover
WHERE Intersects(Buffer(point.geom, 1000), cover.geom)
) result
```

- area(), intersections(), buffer(), within(), overlaps(), reclass(), transform(), centroid(), and many more…

Raster Support Requirements
6 - Storage of Raster Outside of the Database

- **Goals:**
  - **Provide faster access to raster files (JPEGs) for web applications**
  - **Avoid useless database backup of large non-edited datasets**
  - **Avoid importation (copy) of large datasets into the database**



Raster Support Requirements
What about Oracle GeoRaster?

- Stored as a relation between **two types** in different tables:
  - images (**SDO_GEORASTER** for type, extent, rasterTable, id, metadata)
  - blocks (tiles) (**SDO_RASTER** for block information)

images	
id	SDO_GEORASTER
1	type, extent, rasterTable1, id, metadata
2	type, extent, rasterTable2, id, metadata
3	type, extent, rasterTable3, id, metadata
	...

rasterTable3 (blocks)

id	SDO_RASTER
1	id, pyrLevel, band, row, col, MBR, BLOB
2	id, pyrLevel, band, row, col, MBR, BLOB
3	id, pyrLevel, band, row, col, MBR, BLOB
	...

- Supports:
  - bitmap mask
  - two compression schemes
  - three interleaving types
  - multiple dimensions
  - embedded metadata (colour table, statistics, etc…)
  - lots of unimplemented features

- **PostGIS PgRaster** adopts a very similar approach

Raster Support Requirements

Does the Oracle GeoRaster's architecture fulfill our requirements?

Requirement	Yes/No	Comments
1) Non-rectangular raster coverage	Yes but	*Creates as many tables as there are rasters. 1000 rasters = 1000 tables*
2) Easy import/export	No	*Request manual table creation or FME ($$$)*
3) SQL functions & operators on the raster type	Yes	*Although limited*
4) Lossless vector/raster conversion	No	
5) Seamless vector/raster spatial functions/operators	No	*Really not designed for this…*
6) Out-DB Storage	No	

Not really…

PostGIS WKT Raster

An Open Source project specifically designed to meet these requirements

Requirement	Yes/No	Comments
1) Non-rectangular raster coverage	Yes	*Into a single table.*
2) Easy import/export	Yes	*Very similar to PostGIS shp2pgsql.exe & pgsql2shp.exe (gdal2wktraster.py)*
3) SQL functions & operators on the raster type	Yes	*ST_Width(), ST_Height(), ST_BandPixelType(), ST_PixelSizeX(), ST_PixelSizeY(), ST_NumBands(), ST_BandNoDataValue(), ST_GDALGeoTransform(), ST_Resample(), ST_Clip(), ST_Reclass(), ST_MapAlgebra(), ST_AsJPEG(), ST_AsTIFF(), ST_AsPolygon(), etc…*
4) Lossless vector/raster conversion	Yes	*Every raster (or tile) of a single coverage has its own georeference and hence can overlap other rasters.*
5) Seamless vector/raster spatial functions/operators	Yes	*ST_Area(), ST_SRID(), ST_Transform(), ST_Union(), AT_Accum(), ST_AsKML(), ST_AsSVG(), ST_Translate(), ST_Scale(), ST_Intersection(), ST_Intersects(), ST_Within(), ST_PointOnSurface(), &&, etc…*
6) Out-DB Storage	Yes	*Only filepaths are stored in the database.*

PostGIS WKT Raster Status

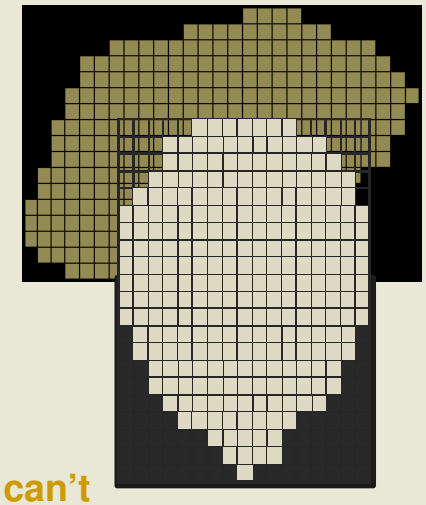
- Contributions
  - Initial code base developed by Sandro Santilli, funded by Steve Cumming (UL, Canada) and Tyler Erickson (Michigan Tech Research Institute)
  - Basic functions, python importer, overviews and regular tiling code: Mateusz Loskot (CadCorp, UK)
  - GDAL Driver foundation: Jorge Arevalo (Google Summer of Code spanish student)
- Version Beta 0.1 to be released soon. Will include:
  - gdal2wktraster.py importer
  - Overviews (multiresolution pyramids) support
  - Accessor Functions (ST_SRID(), ST_Width(), ST_Height(), ST_PixelSizeX(), ST_PixelSizeY(), ST_RotationX(), ST_RotationY(), ST_UpperLeftX(), ST_UpperLeftY(), ST_ESRIWorldFile(), ST_GDALGeoTransform(), ST_NumBands(), ST_BandPixelType(), ST_BandNoDataValue())
  - Basic Seamless Overlay Functions (ST_Intersects(), ST_Intersections(), ST_AsPolygon(), ST_Envelope(), ST_Shape())
  - Spatial operators identical to the one on the geometry type (&&, &<, etc…)
  - Out-DB raster registration with gdal2wktraster.py
  - Well documented web site (doc & wiki specs, http://trac.osgeo.org/postgis/wiki/WKTRaster)
- We also need your help! You can provide developer time or funds…

Introducing
WKT Raster « Raster Objects »

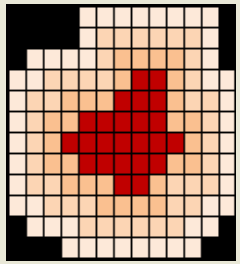
- The fact that every raster in a PostGIS WKT Raster table has its own georeference and attributes, and is thus independent of other rasters in the table, is a very interesting characteric of those raster objects.
- Like vector geometries, raster objects:
  - can overlap
  - can change location
  - can represent individual objects with their own identity
- Moreover, raster objects can be used to model real life objects better represented as small fields (like fires or fuzzy objects).
- Very new type of GIS object

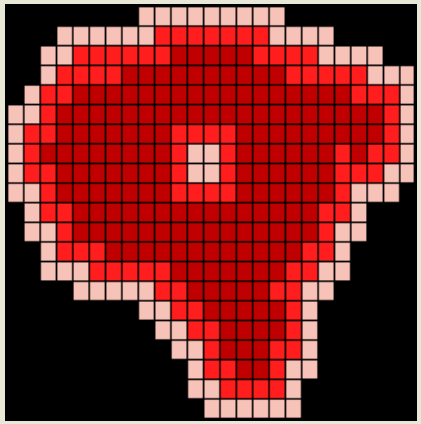
Introducing WKT Raster Objects
Raster Objects vs Other GIS Objects

- Point and Line Coverages
- Polygon Coverages
  - Objects represent a **constant surface** with an **identity and properties** (like an object in a OO context)



- Raster Object Coverages
  - Constant Raster Objects (categorical)
    - Objects represent a **constant surface** with an **identity and properties** (like a feature or an object)
    - Better modelled as **polygon**, but modelled as raster because they are **better processed using existing raster algorithms** (eg. landcover, basin)
    - E.g.: **land use; land cover**; traditional raster objects that should overlap but can't because they are in raster format (ex. **buffers**, **animal territories**)
  - Variable Raster Objects (field)
    - Objects represent a **variable field** that have an **identity and properties**
    - Generally modelised as a unique raster and **difficult to model as polygons**
    - E.g.: **fire**, **fuzzy objects** (**lakes, land cover, forest stands, soil**), **area of influence, animal territories**





- Traditional Raster Coverages
  - Represent a **variable field** with different values (no unique identity or other properties)
  - E.g.: **elevation**, **climate**, etc…

Summary

- The Canadian Spatial Data Foundry should facilitate, via a web service, GIS intersection operations over large-scale ecological datasets (vector & raster)
- Oracle GeoRaster does not provide a good integration between raster and vector layer
- PostGIS WKT Raster aims to provide such an integration
  - Support non-rectangular raster coverages
  - Lossless conversion between raster & vector layers
  - Seamless operators & functions on raster & vector types
  - Storage of raster outside the DB
  - Easy import/export similar to shp2pgsql.exe
  - We need your help!
- WKT Raster introduces a new kind of GIS raster objects that are useful for modelling:
  - categorical features needing raster algorithms
  - fuzzy objects requiring their own identities

Thanks!

- http://trac.osgeo.org/postgis/wiki/WKTRaster