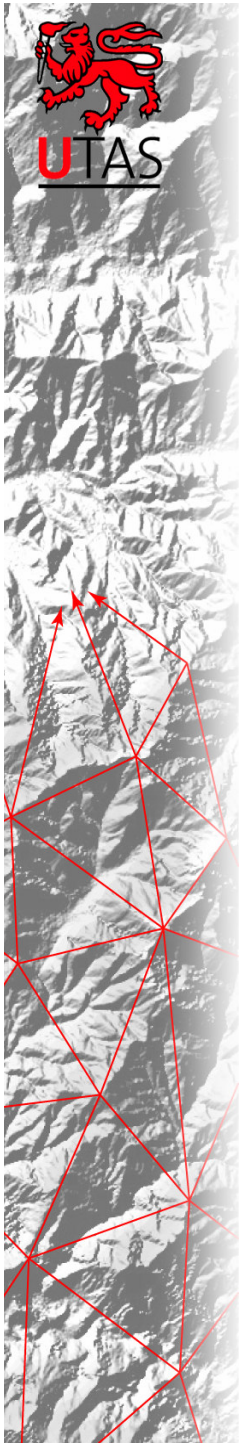


Enterprise “Geo” Databases

Application Development
KGG 380/480

Mr Simon Greener

B.Surv (hons), University of Newcastle.,

Grad. Dip. Comp. Sci, University of Tasmania,
Certificate IV Project Management,

www.SpatialDBAdvisor.com

CenSIS
Centre for Spatial
Information Science

Scare Factor

- Difficulty
  - This is a difficult topic because it is conceptually difficult and I only have 2 hours in which to deliver the material…
  - I liken this lecture to attempting to do a 4 1/2 twisting backwards somersault from the side of the pool.
  - Degree of difficulty: 9**/10**
- But….
- The good news is that: if you can use Microsoft Access or Manifold then this lecture is about the science behind this type of product!
- I might be an Oracle Guru, but as many of my examples as possible will be straight SQL or could be done using Access … except the spatial bits (use Manifold)!

Lecture Outline

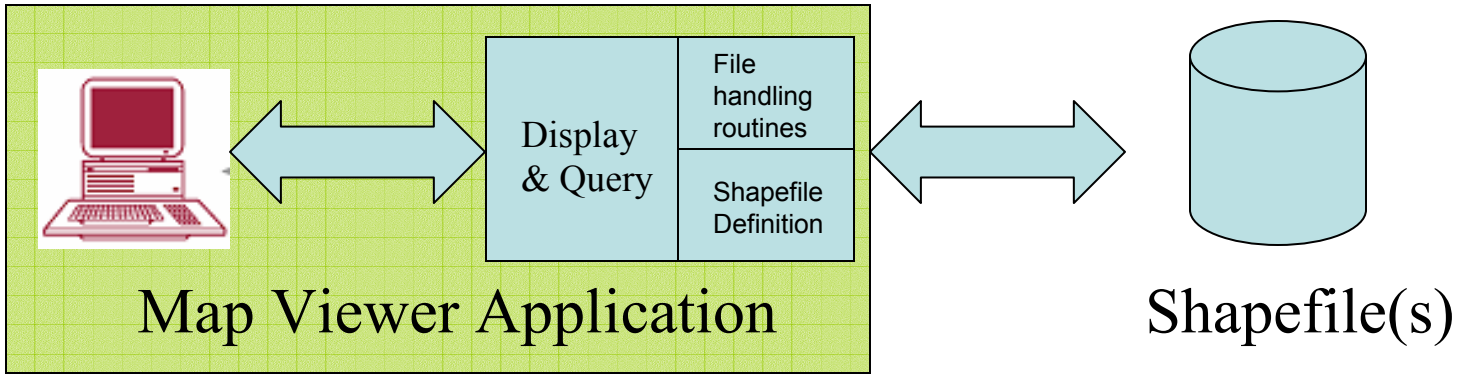
- Data storage
  - Files
  - Databases
- Conceptual modelling by example
- Database implementation models
  - Hierarchical, Network, Relational and Object Oriented
- Spatial data and databases

Data Storage and Computers

- The two main traditional methods.
- Simple Files
  - Containing text or binary in a format that is known only to the application accessing it.
    - Examples
      - ESRI Shapefiles, CAD files, MapInfo TAB files, Manifold Map files
      - GeoTIFFs
      - ESRI Coverages
  - Some are documented “standards” but most are proprietary
    - eg Shapefile physical file format is documented at http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf
- Databases
  - A collection of (binary) files (database) under common, open, management control (DBMS).
  - Applications that use databases do not need to understand how the data is stored in these files.
  - DBMSs: Access, SQLServer, Oracle, Informix IDS, IBM DB2, Postgres. (The internal data structures that these products use are not documented.)

Files - application access

- Applications access files through the following direct “architecture”



Files - Limitations

- Rigid format that is locked to particular application/use.
- No accessible metadata
- Data consistency
  - Attributes in shapefile dbf are not self-checking via integrity rules that are application independent.
  - What is the definition of a correct shape?
  - Bad shapefiles are very common!
- No security.
- Multi-user access limited to read only at best (files are often locked to specific applications).
  - Cannot update a shapefile that is being read by another application!
- Access limited to proprietary systems that understand structure.
  - Not all the shapefile format is published eg spatial index files. Why?
- File format created to satisfy particular functions.
  - New requirements needs new programs to create/access.
  - End user not empowered to modify structure eg topological shapefile!

Databases - Definitions

- **Data**:
  - Known facts that can be recorded and have an implicit meaning.
- **Database**:
  - A collection of interrelated data.
- **Database Management System (DBMS)**:
  - A software package/ system to facilitate the creation and maintenance of a computerized database.
- **Database System**:
  - The DBMS software together with the data itself. Sometimes, the applications are also included.

Key DBMS Features...

- "Self-describing"
  - A DBMS **catalog** stores the *description* (**meta-data**) of the database.
  - Allows DBMS software to work with different databases (sales vs finance vs operations vs planning vs marketing).
- Data Abstraction
  - A **data model** is used to hide storage details and present the users with a *conceptual view* of the database.
- Multiple views of the data
  - User may be presented with views of a database that describe *only* the data of interest to that user.
- Provide multiple levels of security
- Standardised backup and recovery

DBMS Features (2)

- Share data through multiuser transaction processing
  - Allows sets of users to read data from, **and** to update data in, the database.
  - Concurrency control within the DBMS guarantees that each **transaction** is correctly executed or completely aborted.
- Two main types of processing
  - OLTP (OnLine Transaction Processing).
  - OLAP (OnLine Analytical Processing) eg data warehousing.
- Enable DBMS2DBMS Replication
  - Subscriber/Publisher, Master/Slave

DBMS Features (3)

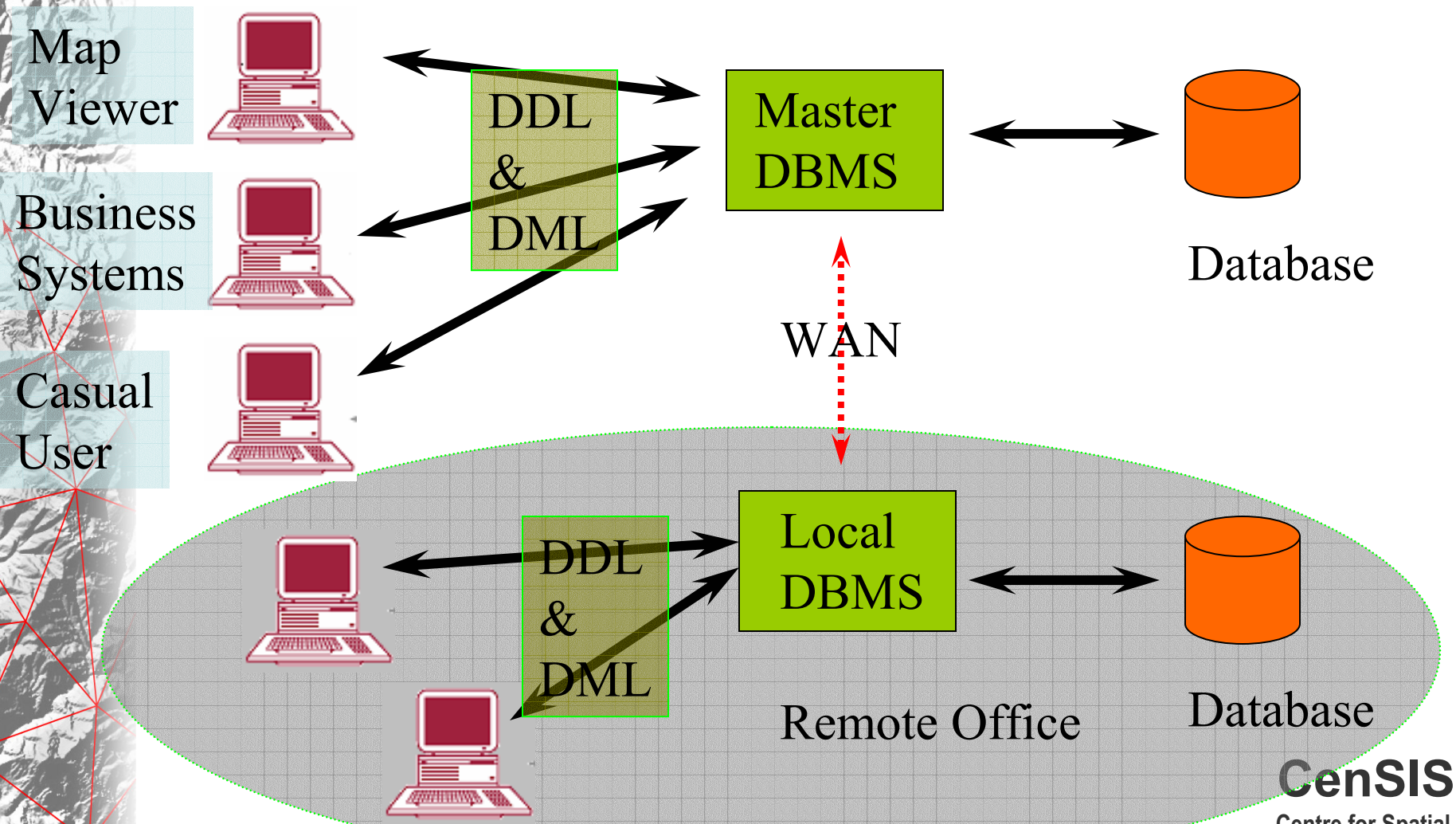
- Insulate programs and data
  - Allows changing data storage structures (new indexing, data partitioning methods) and operations without having to change the DBMS access programs.
- Thus databases:
  - Manage data independently of applications and,
  - Hide physical implementation details from them.

"logical abstraction from physical implementation".

Databases - So how do I access my data?

- Because “how” or “where” the data is stored physically on disk is not known to the application developer or casual user , data is accessed is via “abstracted” languages and APIs.
- These enable a user/application to:
  - Define things through a Data Definition Languages (DDL) ie create objects (tables, queries, indexes etc)
  - Manipulate data through a Data Manipulation Language (DML) ie update, delete and query data.
- These are then implemented in standardised application programmer’s interfaces such as ODBC/OLEDB/JDBC through drivers.

Applications and Databases



DBMS - Advantages

- Minimises Data Redundancy by enabling Data Integration thus providing Data Consistency
- Data Sharing
- Data Independence
- Data Replication
- DDL & DML
- Data Accessibility
- Uniform Security, Privacy and Integrity Controls
- Eases application development
- Reduced program maintenance

Database Disadvantages

- Product Expense
  - Commercial products are expensive (though there are some excellent “open source” databases available eg Postgres, MySQL etc).
- High Overheads
  - Maintenance and Support specialists
  - High end hardware
- Skills
  - Require trained users
- DBMS unnecessary when
  - Database and applications are simple, well defined, and not expected to change.
  - Stringent real-time requirements that may not be met because of DBMS overhead.
  - Multi-user access is not required.
  - Specialist file formats provide for more efficient analysis of static data eg ESRI coverage for overlays for resource analysis

Models of Reality

- But I'm getting ahead of myself…
- All science and thus computing is based on observing the real world, and abstracting it through data models that attempt to represent it.

Model lexicon...

- **Three types of models are pertinent to databases**
  - **Physical** (**low-level**, **internal**)
    - Provide concepts that describe details of how data is stored in the computer. (Files implement this.)
  - **Logical** or **Conceptual** (**high-level**, **semantic**)
    - Provide concepts that are close to the way many users *perceive* data. (Also called **entity-based** or **object-based** data models.)
  - **Implementation** (**representational**)
    - Provide concepts that fall between the above two, balancing user views with some computer storage details.

Logical Modelling ...

- A set of building blocks for (logically) modelling the world is that of "Entities", "Relationships" and "Attributes"
  - *Entity*
    - A person, place, event or concept about which information is recorded.
    - More precisely, an entity is a set or collection of like individual objects called instances. An instance is a single occurrence of a given entity.
  - *Attribute*
    - Describes a characteristic of an entity
  - *Relationship*
    - An association between entities

Relationships...

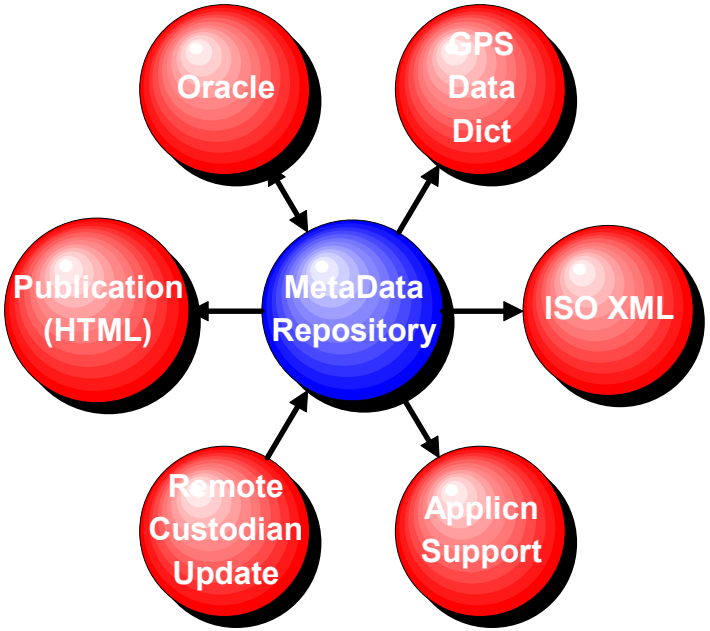
- A relationship is described using the concepts of *degree* and *existence:*
  - *Degree* describes whether the relationship between two entities is 1:1, 1:Many or Many:Many
    eg a pipe is connected to zero, one or more other pipes;
  - *Existence* describes whether entities in a relationship are contingent on each other
    eg water in a pipe may be connected to another pipe; a valve must be connected to a pipe.
- However, with spatial entities, one extra relational property exists: *topology*:
  - A stop valve occurs at the end of a pipe, ie a "meet" relationship between 0-1 objects.

Designing a Model

- Let's show what these things are by a concrete example…
- However, what I am now about to show you is "metadata" - data about data - at work!
  - Managing metadata is the key to good GIS infrastructure
- Computer Aided Design and Engineering (CASE) are database and software "design tools" that make this easier… let's look at Blueprint

Implementation: From Model (metadata) to product...

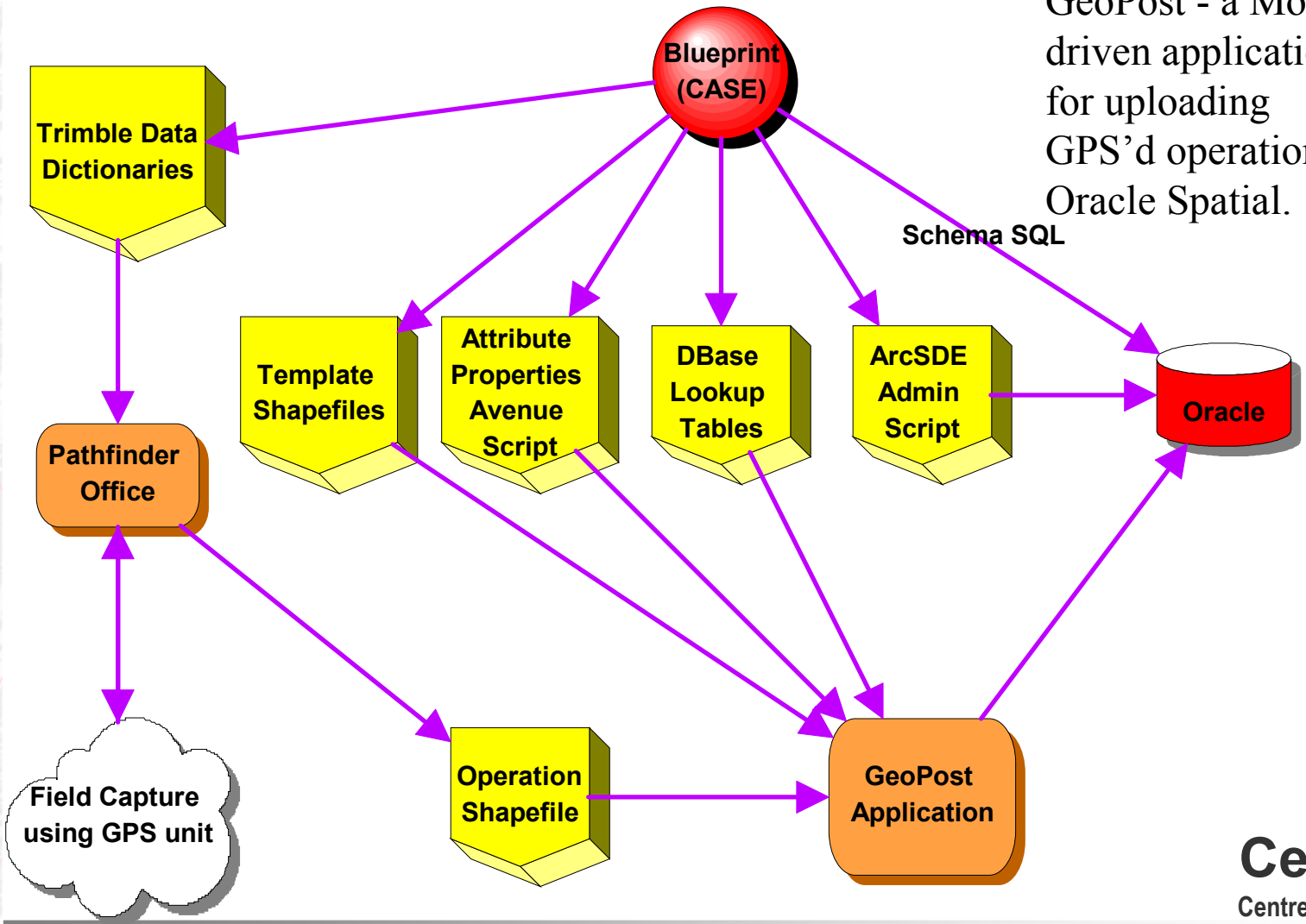
- From logical model in a CASE tool like Blueprint we can:
- *Forward Engineer Products*
  - Create DBMS databases
  - Generate documentation
  - Generate application configuration files (eg GPS data dictionaries)
- *Reverse Engineer*
  - Read DBMS model back into CASE tool.
  - Read table/GIS dataset definitions back into logical CASE tool objects.



CenSIS
Centre for Spatial Information Science

Implementation: From Model to reality via Blueprint

GeoPost - a Model driven application for uploading GPS'd operations to Oracle Spatial.



CenSIS
Centre for Spatial Information Science

Implementation Models: Realising Logical Models in Databases...

- Databases use Implementation Models that represent and manage our logical models in computers.
  - Implemented models can be simple (retrictive) through complicated.
  - Some differ in the method in which relationships (and data methods) are implemented.
  - Main ones are:
    - Hierarchical
    - Network
    - Relational
    - Object-Relational
    - Object Oriented

Implementation Models: History

- Hierarchical Data Model
  - Implemented in a joint effort by IBM and North American Rockwell around 1965. Resulted in the IMS family of systems (and others).
- Network Model
  - First one was implemented by Honeywell in 1964-65 (IDS System).
  - Adopted heavily due to the support by CODASYL (CODASYL - DBTG report of 1971). Later implemented in a large variety of systems - IDMS (Cullinet - now CA), DMS 1100 (Unisys), IMAGE (H.P.), VAX -DBMS (Digital Equipment Corp.).

Implementation Models: History (2)

- Relational Model
  - Proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82.
  - Now in several commercial products (DB2, ORACLE, SQL Server, SYBASE, INFORMIX).
  - Also in “open source” products most notably Postgres (research version of Ingres). MySQL is not a true relational database though it implements the basic model.
- Object-Relational Models
  - Most Recent Trend.
  - Started with Informix Universal Server. Exemplified in the latest versions of Oracle-10i, DB2, and SQL Server etc. systems.

Implementation Models: History (3)

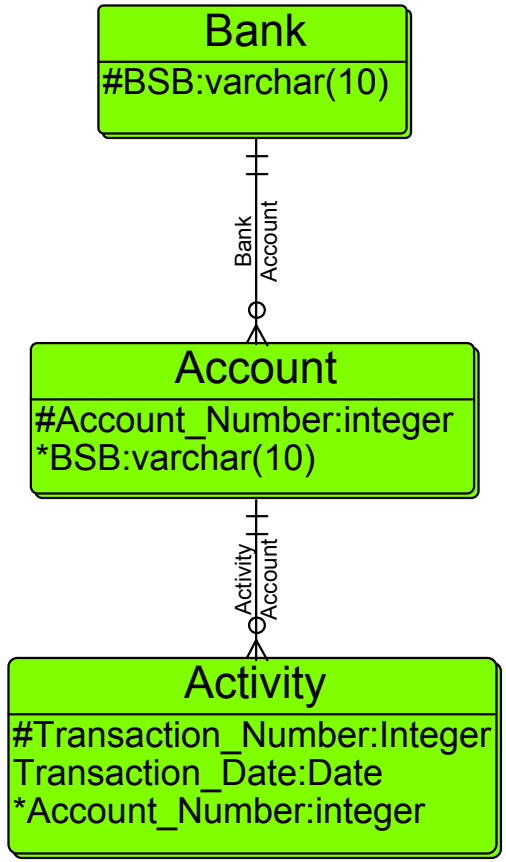
- Object-oriented Data Model(s)
  - Several models have been proposed for implementing in a database system.
  - One set comprises models of persistent O-O Programming Languages such as C++ (e.g., in OBJECTSTORE or VERSANT), and Smalltalk (e.g., in GEMSTONE).
  - Additionally, systems like O_2, ORION (at MCC - then ITASCA), IRIS (at H.P.- used in Open OODB).

Implementation Models: Hierarchical

- The basic building block of the hierarchical model is the "tree" which defines the relationships between entities.
- A "tree" is composed of a single entity called the owner - or root - of the tree linked to zero or more dependent entities, often called members.
- In turn, each member can also be the root of another "tree" of which it is the owner with zero or more dependent members etc.
- However, no member can be connected to (ie owned by) more than one owner (ie the entity immediately above it in the tree).
- When a number of "trees" are linked in this way the entire arrangement can be visualised as an hierarchically arranged structure.
- Relationships are physically implemented through pointers

Hierarchical Databases (2)

- So, in the hierarchical model, the relationships we can use are limited to 1:M.
- While restricted, the hierarchical model does represent some things quite well: A bank, its accounts and their transaction activity; an aeroplane and its passengers; a river network.
- Still used in high-transaction applications (eg ATMs, airline bookings)

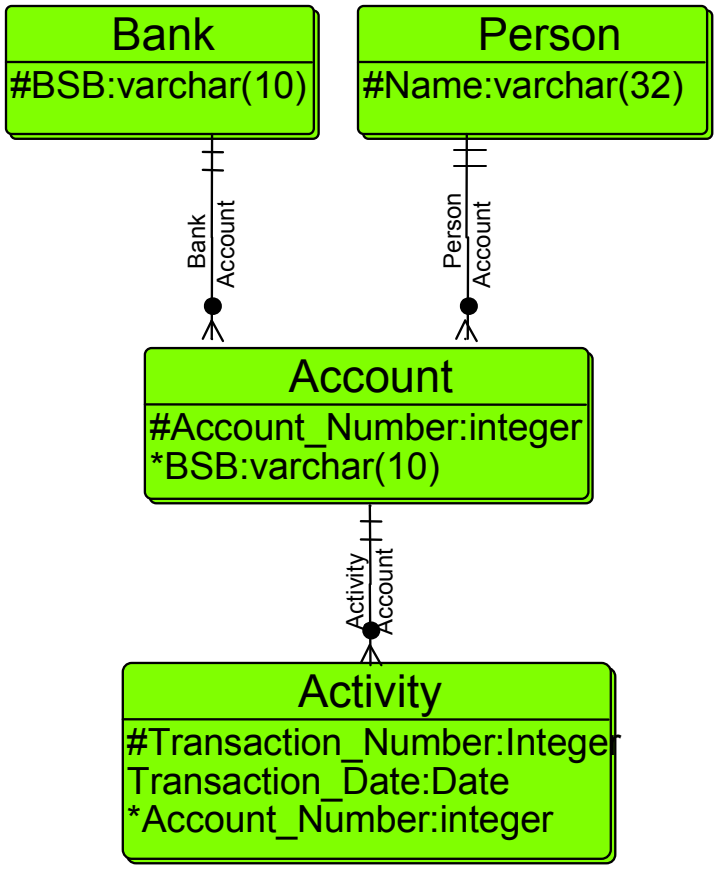


Hierarchical - Summary

- *Advantages*
  - Hierarchical data model is relatively simple and easy to use;
  - Most data processing users, using general experience, are familiar with hierarchies.
  - Because the structure and all relationships are well defined, it is relatively simple to predict performance and capabilities.
- *Disadvantages*
  - "Many to Many" relationships between entities can be implemented only in a clumsy way, and resulting in redundancy in stored data.
  - As a result of the strict hierarchical ordering, the operations of insertion and deletion become unduly complex. (Deletion of an owner results in deletion of its members).
  - The root entity type is always dominant; any member is only accessible through its owner.
  - Hierarchical commands tend to be procedural because of the strictness of the structure.

Implementation Models: Network

- Each record can have multiple parents
  - Composed of sets eg Bank's "Account set" or Persons "Account set"
  - Each set has owner record and member record
  - Member may have several owners



CenSIS
Centre for Spatial Information Science

Network: Summary

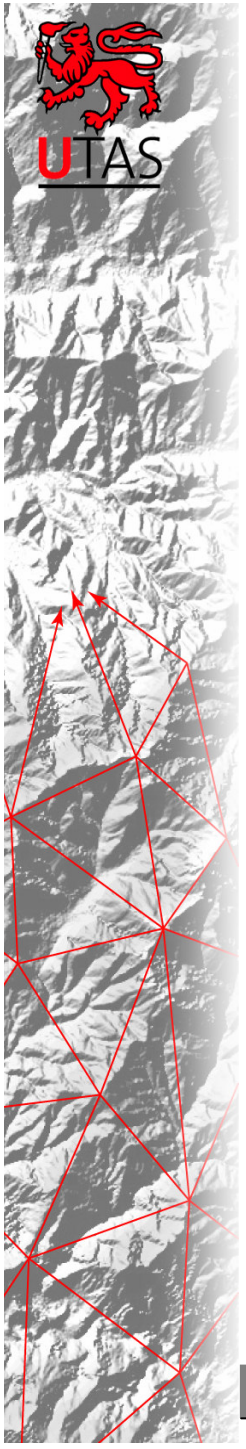
- Advantages
  - Conceptual simplicity
  - Handles more relationship types
  - Data access flexibility
  - Promotes database integrity
  - Data independence
  - Conformance to standards (CODASYL)
- Disadvantages
  - System complexity
  - Lack of structural independence because of navigational and procedural nature of processing

Implementation Models: Relational

- The model was first proposed by Dr. E.F. Codd of IBM in 1970 in the following paper: "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970. (*Earned him the ACM Turing Award.)*
- The relational Model of Data is based on the concept of a Relation.
- A Relation is a mathematical concept based on the ideas of sets.
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations.

Relational: Key Concepts

Informal Terms	Formal Terms
Table	Relation
Column	Attribute/Domain
Row	Tuple
Values in a column	Domain
Table Definition	Schema of a Relation
Populated Table	Extension



Relational Model - Key Concepts (2)

- Tuples (rows) in a relation (table) are **not** considered to be **ordered**; nor is attribute order in tuple.
- All values of attributes in a tuple are considered *atomic* (ie indivisible).
  - A special **null** value is used to represent values that are unknown or inapplicable to certain tuples.
- Constraints are *conditions* that must hold on *all* valid relation instances.
  - Three main types of constraints:
    - **Key** constraints
    - **Entity integrity** constraints
    - **Referential integrity** constraints

Relational Model: Constraints

- Key (single relation)
  - Is a set of attributes that uniquely identify tuples in a relation.
  - A relation can have more than one possible key ⇒ candidate keys. If a relation has *several* candidate keys, one is chosen arbitrarily to be the primary key.
- Entity Integrity (single relation)
  - Check or Column constraints
- Referential Integrity
  - Specifies a *relationship* among tuples in two relations: the **referencing relation** and the **referenced relation**.
  - Tuples in the *referencing relation* R_1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the *referenced relation* R_2.

Relational Model: Diagram

Table name: AGENT

AGENT_CODE	AGENT_LNAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
501	Alby	Alex	B	713	228-1249
502	Hahn	Leah	F	615	882-1244
503	Okon	John	T	615	123-5589

Link through AGENT code

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	05-Apr-2002	502
10011	Dunne	Leona	K	713	894-1238	16-Jun-2002	501
10012	Smith	Kathy	W	615	894-2285	29-Jan-2001	502
10013	Olowski	Paul	F	615	894-2180	14-Oct-2002	502
10014	Orlando	Myron		615	222-1672	28-Dec-2002	501
10015	O'Brian	Amy	B	713	442-3381	22-Sep-2002	503
10016	Brown	James	G	615	297-1228	25-Mar-2002	502
10017	Williams	George		615	290-2556	17-Jul-2002	503
10018	Farriss	Anne	G	713	382-7185	03-Dec-2002	501
10019	Smith	Olette	K	615	297-3809	14-Mar-2002	503

CenSIS
Centre for Spatial Information Science

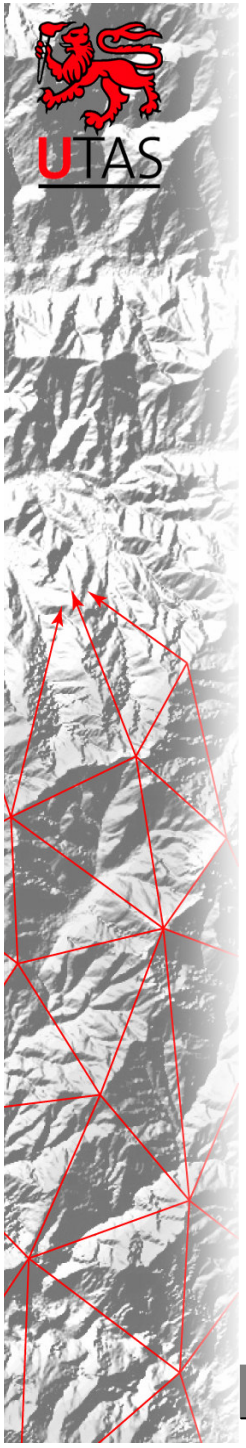
Relational Model: SQL

- Structured Query Language
  - Non-procedural language for defining and manipulating relations.
  - For example on previous slide:

```
Select   cus_fname, cus_initial,
         cus_lname, cus_phone
 from    agent,
         customer
where (          agent.agent_fname = 'John'
         and     agent.agent_initial = 'T'
         and     agent.agent_lname = 'Okon' )
  and customer.agent_code = agent.agent_code
```

Relational Model: Summary

- Advantages
  - Formal mathematical theory
  - Structural independence
  - Improved conceptual simplicity
  - Easier database design, implementation, management, and use
  - Ad hoc query capability with SQL
  - Powerful database management system
- Disadvantages
  - Substantial hardware and system software overhead
  - Poor design and implementation is made easy
  - May promote “islands of information” problems
  - Difficulties with strict relational model as applied to complex objects like vector shapes!

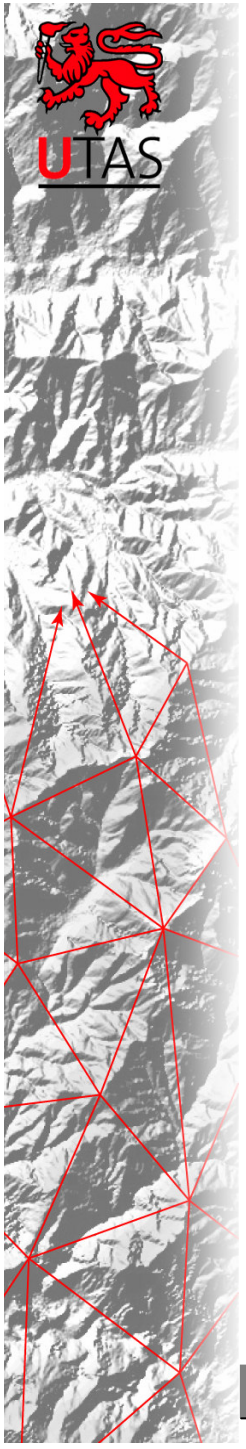


Object Relational by example: “Geo” Databases

- All data is corporate data and should be held within a single database and accessible by all.
- However, databases and DBMS have only stored and manipulated traditional data types: numbers, text, dates [Point data are often stored in (relational) databases in 2 numeric columns: (Long,Lat) or (Easting,Northing)]
- But what about non-traditional data types like Spatial data?
  - ☹ Hierarchical/Network (too old, no interest)
  - ☺ OODBs (by their very nature)
  - 😐 Partially possible in traditional DBMS through “normalised” models (see OpenGIS Simple Features for SQL) or proprietary methods like ArcSDE “ESRI Binary”
  - ☺ Definitely possible in “Object” extended RDBMS (O-RDBMS)

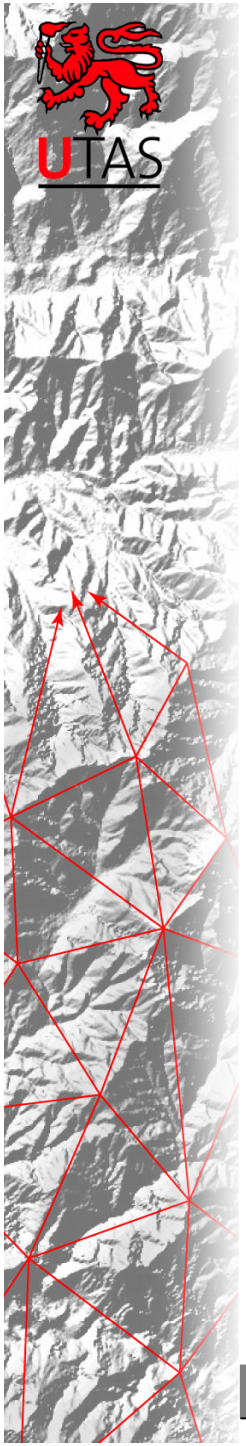
Object Relational by example: “Geo” Database

- What types of “Spatial”?
  - Any type for which an Abstract Data Type can be defined!
- So, notion of “Abstract Data Types” ADT is key to supporting non-traditional data types.
- ADT is data and methods (an Object as in OOP)
  - Polygon shape is coordinates defining it + methods that can operate on it eg New (Polygon), Insert/Update/Delete (coordinate), Area, Perimeter etc
- But also need to define topological operations so that relational concepts like referential integrity can be implemented in (O)RDBMS. But FK/PK is more than just Equals it can be inside, meets (touches), etc.



Fundamental (vector) objects

- 0-D objects: these are *points*. These have a boundary only.
- 1-D objects: these are *lines*. These have a boundary which consists of the two endpoints of the line, and an interior which is the rest of the line. A line which joins up with itself does not have any endpoints and hence no boundary.
- 2-D objects: these are *polygons*. They have a boundary (around the edge) and an interior.



Boundary/Interior intersections

∂∂: Boundary of A intersects with Boundary of B.

°°: Interior of A intersects with Interior of B.

∂°: Boundary of A intersects with Interior of B.

°∂: Interior of A intersects with Boundary of B.

Topological Relationships

- Given objects and intersections we can define topological relations as ⇒
- SQL Extensions implement.

∂∂	°°	∂°	°∂	0-0	0-1	0-2	1-1	1-2	2-2
0	0	0	0						
1	0	0	0						
0	1	0	0						
1	1	0	0						
0	0	1	0						
1	0	1	0						
0	1	1	0						
1	1	1	0						
0	0	0	1						
1	0	0	1						
0	1	0	1						
1	1	0	1						
0	0	1	1						
1	0	1	1						
0	1	1	1						
1	1	1	1						

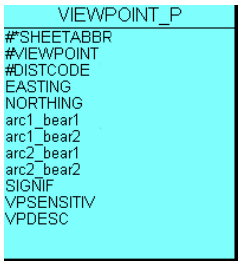
CenSIS
Centre for Spatial Information Science

Files vs Databases Revisited: SDEBINARY

- ESRI's ArcSDE supports two in-database formats: Oracle Spatial (object) native format and SDEBINARY.
- Oracle recommends SDO_GEOMETRY (Oracle's native spatial data type) over using a LONG RAW data type.
- Storing spatial data in SDO_GEOMETRY compromises no ESRI product features.
- ESRI SDEBINARY recommendations almost always refer to the LONG RAW data type storage format in Oracle.

Files vs Databases Revisited: SDEBINARY (2)

- SDEBINARY is a proprietary format with some of the disadvantages previously noted for files.
- SDEBINARY is organised on three levels:
  1. **Three** Oracle tables compose a **single** logical entity.
     Eg. viewpoint_p, shape - Fnnn and spatial index - Snnn (linked through fid primary key/foreign key)
  2. "points" attribute in Fnn table (LONG RAW) in which shape's points are organised in a proprietary manner not visible to non-ESRI applications.
  3. Single "shape" (Fnnn table) per business entity.



CenSIS
Centre for Spatial Information Science

Oracle Spatial (OS) vs (LONG RAW) SDEBINARY

- Unrestricted Modelling Options.
  - With OS, users can model and store business data in any way they want eg a user can use the more natural single table representation rather than three table SDEBINARY approach.
  - OS allows multiple shape columns per business entity (can only be accessed in ArcSDE via single SDO_GEOMETRY views).
  - OS imposes no restrictions on use of Views, Materialized Views, cross-database links: SDEBINARY views must be created through ESRI tools and restricts use of Oracle features.

OS vs SDEBINARY (2)

- (None of the features that follow are available when using the LONG RAW data type.)
- Partitioning
  - Key feature for a truly scalable solution.
  - Both Oracle tables and (SDO_GEOMETRY) spatial indexes can be partitioned.
  - Partitioning improves manageability and performance.

OS vs SDEBINARY (3)

- Spatial Indexing
  - Spatial indexes are available for both vector and raster spatial data types.
  - For all insert/update/deletes on SDO_GEOMETRY data, Oracle automatically keeps spatial index current (with SDEBINARY all Inserts/updates/deletes have to be controlled by an external process).
  - Spatial index operations can also leverage parallel processing options.

OS vs SDEBINARY (4)

- Spatial Operators are available via SQL
  - topological query
  - within distance
- Spatial Functions via SQL
  - linear referencing,
  - dynamic segmentation,
  - coordinate system transformations,
  - geometry analysis,
  - raster data and geometry aggregation

OS vs SDEBINARY (5)

- Stored Procedures, Functions and Triggers
  - Enables “in database” scripting of spatial data eg implement spatial integrity rules such as topological relationships (stop valve must fall at the end of a pipe)
- Data Validation
  - Spatial data can be checked for integrity in the database eg GPS data directly written to DBMS from GPS software (such as Trimble Pathfinder Office) can be checked for quality/integrity.

OS vs SDEBINARY (6)

- Persistent Topology
  - Choice of topological vs non-topological data models for storing data in Oracle Spatial both of which are available to any client/GIS.
- Whole Earth Coordinate Systems
  - OS can store data in any coordinate system/datum.
  - No limitation on size of coordinates
  - SDEBINARY coordinates limited to signed long integers (max range 0 - 2.1 billion ie 2^{31})

OS vs SDEBINARY (7)

- Replication
  - Tables with SDO_GEOMETRY data can be replicated by standard Oracle replication services
    - one way (master-slave),
    - two-way (multiple master) or
    - n-way (3 or more nodes).
  - Tables with SDEBINARY cannot be replicated.

OS vs SDEBINARY (8)

- Spatial Driven Multi-Level Security
  - Oracle's Virtual Private Database (VPD) technology can be used by administrators to write security policies using Oracle PL/SQL even for SDO_GEOMETRY (not SDEBINARY)

OS vs SDEBINARY (final)

- Partner Inter-operability in Server
  - SDO_GEOMETRY is the only data type in Oracle natively integrated by all the major GIS and GeoSpatial Application vendors.
  - ESRI's LONG RAW alternative integrates ESRI applications ***only***.

Example: Live Fire Mapping

- As a new fire boundary (created by flying a helicopter around the fire edge) is inserted into the database clip out the tenure/veg that has been burned and insert it into a table so that it can be viewed in a system built using Apache Struts:

```
SELECT /*+ORDERED ORDERED_PREDICATES */
           a.OPERATION_TYPE_ID,
           a.OPERATION_ID,
           a.BURN_DATE,
           b.TENURE_CODE,
          b.VEGETATION_TYPE_CODE,

ROUND(SUM(MDSYS.SDO_GEOM.SDO_AREA(MDSYS.SDO_GEOM.SDO_INTERSECTION(b.shape,c_diminfo,a.shap
e,c_diminfo),c_diminfo)) / 10000,2)
       FROM ops.burn_daily_incident a,
            ops.fire_tenveg b
      WHERE a.featureid = c_featureid
        AND MDSYS.SDO_RELATE(b.shape,a.shape,'mask=ANYINTERACT querytype=WINDOW') = 'TRUE'
        AND (MDSYS.SDO_GEOM.SDO_INTERSECTION(b.shape,c_diminfo,a.shape,c_diminfo) is not
null
        AND
ROUND(MDSYS.SDO_GEOM.SDO_AREA(MDSYS.SDO_GEOM.SDO_INTERSECTION(b.shape,c_diminfo,a.shape,c_
diminfo),c_diminfo),2) > 0.0)
     GROUP BY a.OPERATION_TYPE_ID,
              a.OPERATION_ID,
              a.BURN_DATE,
              b.TENURE_CODE,
              b.VEGETATION_TYPE_CODE;
```

Live Fire Mapping (cont)

- This SQL statement is hosted inside an Oracle pl/sql procedure that is fired off through an AFTER INSERT trigger on the table that accepts the fire boundary (polygon).
- The firing occurs by placing the request into the Oracle queue :

```
job_string := 'ops.BurnTenureVegtype('||to_char(v_featureid)||','''||USER||''');';
DBMS_JOB.SUBMIT(jobno,job_string,SYSDATE,NULL);
```

- The request to the DBMS_JOB makes the processing asynchronous to the actual insert of the fire shape. This is important because the clip out can take a bit of time when the shape is large... but mostly these clip outs take < a few seconds! When the procedure completes it sends an email to the person who put in the fire boundary with its performance statistics (Yes, straight out of the database!). An example of an email is:

```
Operation: 12345
Description: Burn area computations completed for recently updated burn area shape.
Records: 15 written to burn_area table.
Processing Time: 0 minutes 9 seconds.
Action: Please use FIRES application to view tenure/veg/area values.
```

Property Rights (1)

- FT creates many dealings over state forest: leases, licenses, easements etc. Some of these dealings can be quite complicated. They also have some important vertical topological requirements (for example: a dealing of a particular type MUST fall wholly within state forest).

Property Rights (2)

- Requirement:
- SQL

```
select CASE WHEN
   round(mdsys.sdo_geom.sdo_area(pr1.shape,a.diminfo)) <>

round(sum(mdsys.sdo_geom.sdo_area(mdsys.sdo_geom.sdo_intersection(pr2.shape,a.diminfo,pr1.shape,a.diminfo),a.dim
info)))
  THEN 'RULENOK'
  ELSE 'RULEOK'
  END AS RULE
FROM all_sdo_geom_metadata a,
     propright.property_right pr1,
     propright.property_right pr2
WHERE ( a.owner = 'PROPRIGHT' AND a.table_name = 'PROPERTY_RIGHT' and a.column_name = 'SHAPE')
  AND ( pr1.property_right_id =  :PR1 )
  AND ( mdsys.sdo_relate(pr2.shape,pr1.shape,'mask=ANYINTERACT querytype=window') = 'TRUE'
        and
        pr2.property_right_id <> :PR1
        AND pr2.property_right_type = :PT2 )
group by round(mdsys.sdo_geom.sdo_area(pr1.shape,a.diminfo))
```

Property Rights (2)

- Requirement:
- SQL:

```
select pr2.property_right_id,
       pr2.property_right_status,

substr(mdsys.sdo_geom.relate(pr2.shape,a.diminfo,'mask=DETERMINE',pr1.shape,a.diminfo),1,2
0) as TopoRelnShp
FROM all_sdo_geom_metadata a,
     propright.property_right pr1,
     propright.property_right pr2
WHERE ( a.owner = 'PROPRIGHT' AND a.table_name = 'PROPERTY_RIGHT' and a.column_name =
'SHAPE')
  AND ( pr1.property_right_id =  :PR1 )
  AND ( mdsys.sdo_relate(pr2.shape,pr1.shape,'mask=ANYINTERACT querytype=window') = 'TRUE'
        and
        pr2.property_right_id <> :PR1
        and
        pr2.property_right_type = :PT2 )
```

Land Property (2)

- We have implemented all these integrity rules via simple SQL statements that are used both in our Property Rights application AND within the database.
- No GIS software is needed other than the Oracle database.

Object Databases: Advantages

- Objects **do not require re-assembling** from their component tables each time they are used thereby reducing processing overheads by increasing access speeds
- **Paging** is reduced
- **Versioning** is easier
- **Navigation** through the database is easier and more natural, with objects able to contain pointers to other objects within the database
- **Reuse** reduces development costs
- **Concurrency control** is simplified by the ability to place a single lock on an entire hierarchy (while still retaining the choice to lock individual objects)
- **Better data model** as based on the 'real world' instead of the 'flattened' relational model
- Good for applications where the ***relationships*** between items in the database carry **key information** *e.g. in a student database, we were particularly interested in what students studied. This is handled very efficiently by navigation.*

Object Databases: Disadvantages

- Poor for applications where the ***values*** of items in the database carry **key information** *e.g. if we had been more interested in student age (e.g. to calculate the mean age) than the courses they study then relational database would clearly be more efficient*
- **Speed of access** may be reduced by late binding which may cause extensive searches through the inheritance hierarchies
- Present **lack of standards** including the lack of a common query language such as SQL (though OQL on its way?)
- There are as yet **no formal semantics** for ODBMS. Relational databases can be 'proved' correct by means of set theory and relational calculus
- The simplicity of **relational tables** is lost
- The object oriented **paradigm shift** can make the move to ODBMS difficult

Standards, standards, standards...

- We need is for the relevant ISO/TC (eg 211) and other SQL (eg SQL3/MM) standards to move forward to such an extent that we can execute DML and DDL against any commercial database using a standard dialect.
- Once we have this we need the relevant ODBC/OLEDB/ADO/JDBC etc drivers to support a truly standardised set of spatial data types.