Database Systems
The engine for GIS
Supports data storage and sharing, allows data to be modified and analyzed in the store
Characteristics of a database system
- Quality control assurance and security
- Flexible access for different classes of users
- Flexible methods for retrieval
- Possibility to link pieces of information
- Possibilities for concurrent access by multiple users

Data Models
Collection of constructs for describing and structuring applications in the database

For Developers
- Represents the application domain in a way that it can be translated into a system design and implementation

For Users
- Provides a description of the structure of the system independent of the data or details of the implementation

Semantic data model
- Represents the meaning of the application domain as closely as possible

Implementation models
- Record-based, object-based, object-relational

Human Database Interaction

Data definition
- Description of the conceptual and logical organization

Storage definition
- Description of the physical structure, file location and indexing

Database administration
- Daily operation of the database

Data manipulation
- Insertion, deletion, modification and retrieval of data from the database
Database Management

DBMS – software system that manages the database

- The logical unit of interaction with the DBMS is the transaction
  Transaction: Create, modify, delete
- Transaction are either executed in entirety (committed) or rolled back
- On commit, all changes since last commit are made permanent
- Maintains ACID properties - atomicity (all or nothing), consistency, isolation (no side effects) and durability (can survive a crash).

Relational Data Models

Relational database is a collection of tabular relations containing a fixed set of fields or attributes

- The data in a relation are structured as a set of rows
- A row or tuple consists of a list of values, one for each attribute
- The primary key is an attribute that uniquely identifies a record
- An attribute is associated with a domain – its range or valid values
- Relational databases require atomic values

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute1</td>
<td>Well-ID</td>
<td>Numeric</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Attribute2</td>
<td>Depth</td>
<td>Float</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Attribute3</td>
<td>pH</td>
<td>Float</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Attribute4</td>
<td>Town</td>
<td>Character</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

Relation

<table>
<thead>
<tr>
<th>Well_ID</th>
<th>Depth</th>
<th>pH</th>
<th>Town</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>250.5</td>
<td>6.54</td>
<td>Frankfort</td>
</tr>
<tr>
<td>215</td>
<td>540.6</td>
<td>6.8</td>
<td>Bucksport</td>
</tr>
<tr>
<td>340</td>
<td>467.8</td>
<td>7.1</td>
<td>Dixmont</td>
</tr>
</tbody>
</table>
Relational Database Operations

Union: (∪) builds the set-theoretic union of two relations. Given relations R and S (both must have the same arity), the union $R \cup S$ is the set of tuples that are in R or S or both.

Intersection: (∩): builds the set-theoretic intersection of two relations. Given the tables R and S, $R \cap S$ is the set of tuples that are in R and in S. R and S must have the same arity.

Difference: (−): builds the set difference of two relations. Let R and S be two tables with the same arity. $R - S$ is the set of tuples in R but not in S.

Select: (σ): extracts tuples from a relation that satisfy a given restriction.

Project: (π): extracts specified attributes (columns) from a relation.

Join: (⋈): connects two relations by their common attributes.
Relational Database Interaction and SQL

SQL – Structured Query Language – specialized query language for interaction with relational databases
- Serves as a data definition language (DDL)
- DDL used to create, modify or delete a relation schema

CREATE TABLE DEPARTMENTS (  
DEPT_ID NUMBER(10) NOT NULL PRIMARY KEY,  
NAME VARCHAR2 (50) NOT NULL)

Data Manipulation using SQL

Use of SQL as data manipulation language for data retrieval

SELECT *  
FROM Wells  
WHERE pH > 7
- The select clause indicates an attribute to be retrieved from a relation – a * indicates all attributes
- From is used to indicate from which tables the data is to be taken, as well as how the tables JOIN to each other.
- The where clause indicates the restrict condition

Data Manipulation using SQL

Relational joins are allowed by calling more than one relation in the from clause

Example: find wells that serve towns with populations greater than 10000

SELECT Wells.town  
FROM Wells, Towns  
WHERE Wells.town = Towns.town and population > 10000

Relational Databases and Geographic Information

Two approaches
- Integrated - Put all data, spatial and non-spatial, in the relational database (Smallworld)
- Hybrid - Store only the non-spatial data in the database and the spatial data in separate files (Arc/INFO)

Problems with Integrated approach
- Slow retrieval due to multiple joins
- Inappropriate indexes and access methods
- SQL not effective for spatial queries
Spatial data in relational table

Census Blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Area</th>
<th>Population</th>
<th>Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1020</td>
<td>2</td>
<td>2395</td>
<td>Polygon((1,5), (5,5),(5,1),(1,5))</td>
</tr>
</tbody>
</table>

What you would like to do.

Census Blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Area</th>
<th>Population</th>
<th>Boundary-ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1020</td>
<td>2</td>
<td>2395</td>
<td>1020</td>
</tr>
</tbody>
</table>

Edges

<table>
<thead>
<tr>
<th>Edge_name</th>
<th>Endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

Points

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>X-coord</th>
<th>Y-coord</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Four tables required to store spatial data in a relational model

SQL not designed for Spatial Queries

```
SELECT * 
FROM Wells 
WHERE pH > 7 

SELECT Name 
FROM Census_blocks A,B 
WHERE adjacent(A,B) and A.name = 100
```

Object-based Data Models

Based on objects or entities

Something with an independent and uniquely identifiable existence in the application domain

Entity- Relation -Attribute Model

- Entities are described by their attributes
- Entities have explicit relations with other entities
- Entities are grouped into entity types – those of the same type have the same attributes and relationship structures
Object-oriented Approach

- OO approach is used both as a method of **semantic data modeling** and as model for data handled by OO programming and database management.
- OO model adapts OO programming constructs to database management systems.

Object-oriented Constructs

- **Encapsulation** – creates an identifiable collection of data and code (methods) that operate on the data.
- **State** of an object is determined by the data items within its wrapper.
- **Instance variables** – wrapped data items:
  - Values of the data items are themselves objects.
  - Objects invoke their own methods.
  - Objects send messages to invoke the methods of other objects.
  - Objects thus have state and behavior.

Object Classes: Well and Point

- **Well**
  - ID: Integer
  - Depth: Float
  - pH: Float
  - Town: String
  - Extent: Point

- **Point**
  - ID: Integer
  - X Coord: Float
  - Y Coord: Float

/Update pH:pH→pH

Point used as instance variable by class Well.

Inheritance

- **Inheritance** – creation of a new class of object by modification of an existing class.
- A subclass inherits instance variables and methods of the originating or superclass.
- Subclasses may have additional instance variables and specialized methods.

**Operator polymorphism**: operator with the same name has different implementations for different classes.

- Polygon
- Triangle
- Octagon

[Diagram of inheritance relationships with classes Polygon, Triangle, and Octagon, and methods Area and Update pH.]
### Composition of Objects

**Composition** - allows modeling of complex structures

- **Aggregation** – collection of objects from different classes into an aggregate class
  - Runway, terminal, hanger, apron → airport

- **Association** – groups objects of the same class into an associated class
  - Towns → region

### Object Oriented Database Management Systems

**Support:**
- Persistent objects, object classes, and inheritance hierarchies
- Non-procedural query languages for object class definition, manipulation and retrieval
- Efficient query handling
- Appropriate transaction processing

**Two Approaches**
- Extend relational to handle OO
- Build database around OO programming language

### Object Oriented Database Management Systems

#### Creating persistent objects – Class → Persistent Object

**Methods**
- Create new persistent object
- Delete persistent object
- Retrieve the state of a persistent object
- Modify a persistent object

**Methods**
- RDM - Call by value
- OODB - Call by reference
- Match values
- Navigate by OIDs

### Object - Relational

**Enhance relational model with object-oriented features**
- Allows complex – user defined data types
- Inheritance, aggregation and object identity
- SQL3 adds support for objects
- Maintains efficient performance of relational model
- Objects in the relational model are tuples of atomic values
- Object – relational systems allow non-atomic types
- Nested relations – value of an attribute is a relation
- Support of indexes for user defined data types
**SQL3**

Provides the basis for supporting object-oriented structures

- user-defined types (ADTs, named row types, and distinct types)
- type constructors for *row types*
- type constructors for *collection types* (sets, lists, and multisets)
- user-defined functions and procedures
- support for *large objects* (BLOBs and CLOBs)

```sql
CREATE OR REPLACE TYPE PERSON_T AS OBJECT
  (PERSON_ID NUMBER(10),
   LNAME VARCHAR2 (50),
   FNAME VARCHAR2 (50),
   BIRTH_DATE DATE)

CREATE OR REPLACE TYPE EMP_T AS OBJECT
  (EMP_ID NUMBER(10),
   PERSON PERSON_T,
   HIRE_DATE DATE)

CREATE TABLE EMP OF EMP_T
  (EMP_ID NOT NULL PRIMARY KEY)
```

---

**OOGIS**

- Allows definition of spatial objects and inheritance hierarchy
- Allows definition of methods on these classes
e.g. Area, distance, intersection, union

---

**Oracle Spatial Data Model**

A hierarchical structure consisting of elements, geometries, and layers.

**Element**
The basic building block of a geometry. The supported spatial element types are points, line strings, and polygons.

**Geometry or geometry object**
The representation of a spatial feature, modeled as an ordered set of primitive elements. It can consist of a single element, or a homogeneous or heterogeneous collection of elements.

**Layer**
A collection of geometries having the same attribute set.
Oracle Spatial defines the object type SDO_GEOMETRY as:

CREATE TYPE sdo_geometry AS OBJECT(
  SDO_GTYPE NUMBER,  \(d\) digit number in format \(dltt\), eg. 2003
  SDO_SRID NUMBER,  \(d\) identifies the number of dimensions (2, 3, or 4)
  SDO_ELEM_INFO MDSYS.SDO_ELEM_INFO_ARRAY,  \(l\) identifies the linear referencing measure dimension for a three-dimensional linear. For a non-LRS geometry the value is 0.
  SDO_ORDINATES MDSYS.SDO_ORDINATE_ARRAY);

\(t\) identifies the geometry type (00 through 07, with 08 through 99 reserved for future use)

<table>
<thead>
<tr>
<th>Value</th>
<th>Geometry Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>UNKNOWN_GEOMETRY</td>
<td>Spatial ignores this geometry.</td>
</tr>
<tr>
<td>01</td>
<td>POINT</td>
<td>Geometry contains one point.</td>
</tr>
<tr>
<td>02</td>
<td>LINE or CURVE</td>
<td>Geometry contains one line string that can contain straight or circular arc segments, or both. (LINE and CURVE are synonymous in this context.)</td>
</tr>
<tr>
<td>03</td>
<td>POLYGON</td>
<td>Geometry contains one polygon with or without holes.</td>
</tr>
<tr>
<td>04</td>
<td>COLLECTION</td>
<td>Geometry is a heterogeneous collection of elements.</td>
</tr>
<tr>
<td>05</td>
<td>MULTIPOLYGON</td>
<td>Geometry has one or more points. (MULTIPOLYGON is a superset of POINT.)</td>
</tr>
<tr>
<td>06</td>
<td>MULTILINE or MULTICURVE</td>
<td>Geometry has one or more line strings. (MULTILINE and MULTICURVE are synonymous in the context, and each is a superset of both LINE and CURVE.)</td>
</tr>
<tr>
<td>07</td>
<td>MULTIPOLYGON</td>
<td>Geometry can have multiple, disjoint polygons (more than one exterior boundary). (MULTIPOLYGON is a superset of POLYGON.)</td>
</tr>
</tbody>
</table>

Oracle Spatial Data Model

SDO_TYPE  Indicates the type of the geometry

SDO_SRID  used to identify a coordinate system (spatial reference system) to be associated with the geometry.

SDO_POINT  defined using the SDO_POINT_TYPE object type, which has the attributes X, Y, and Z, all of type NUMBER.
Oracle Spatial Data Model

**SDO_ELEM_INFO**

defined using a varying length array of numbers. This attribute interprets the ordinates stored in the SDO_ORDINATES attribute.

- **SDO_STARTING_OFFSET** -- Indicates the offset within the SDO_ORDINATES array where the first ordinate for this element is stored. Offset values start at 1 and not at 0. Thus, the first ordinate for the first element will be at SDO_GEOMETRY.SDO_ORDINATES(1). If there is a second element, its first ordinate will be at SDO_GEOMETRY.SDO_ORDINATES(n), where n reflects the position within the SDO_ORDINATE_ARRAY definition.

- **SDO_ETYPE** - Indicates the type of the element. SDO_ETYPE values 1, 2, 1003, and 2003 are considered simple elements. They are defined by a single triplet entry in the SDO_ELEM_INFO array. For SDO_ETYPE values 1003 and 2003, the first digit indicates exterior (1) or interior (2):
  - 1003: exterior polygon ring (must be specified in counterclockwise order)
  - 2003: interior polygon ring (must be specified in clockwise order)

Example: SDO_Geometry for a rectangle

- **SDO_GTYPE** = 2003. 2 indicates two-dimensional, 3 indicates a polygon.
- **SDO_SRID** = NULL.
- **SDO_POINT** = NULL.
- **SDO_ELEM_INFO** = (1, 1003, 3). The final 3 in 1003,3 indicates that this is a rectangle. Because it is a rectangle, only two ordinates are specified in SDO_ORDINATES (lower-left and upper-right).
- **SDO_ORDINATES** = (1, 1, 7, 5). These identify the lower-left and upper-right ordinates of the rectangle.

CREATE TABLE Bank_Mket (
  bank_id NUMBER PRIMARY KEY,
  name VARCHAR2(32),
  shape MDSYS.SDO_GEOMETRY);

INSERT INTO Bank_Mket VALUES (1, 'bank_a',
  MDSYS.SDO_GEOMETRY(2003, -- 2-dimensional polygon
  NULL,
  NULL,
  MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3), -- one rectangle (1003 = exterior)
  MDSYS.SDO_ORDINATE_ARRAY(1, 1, 7, 5) -- only 2 points needed to define rectangle (lower left and upper right))

Oracle Spatial Data Model

**SDO_ORDINATES**

a varying length array of NUMBER type that stores the coordinate values that make up the boundary of a spatial object. This array must always be used in conjunction with the SDO_ELEM_INFO varying length array.
Further Challenges

Addressing the uncertainty in spatial objects

Addressing change – dynamic qualities

- Temporal databases
  Stores snapshots

- Dynamic databases
  Supports evolving snapshots