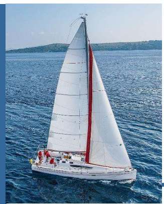
فصل هشتم: انواع داده پیچیده

(Complex Data Types)

درس پایگاه داده
دانشگاه صنعتی نوشیروانی بابل
مهدی عمادی
m.emadi@nit.ac.ir



Semi-Structured Data

- Many applications require storage of complex data, whose schema changes often
- The relational model's requirement of atomic data types may be an overkill
  - E.g. storing set of interests as a set-valued attribute of a user profile may be simpler than normalizing it
- Data exchange can benefit greatly from semi-structured data
  - Exchange can be between applications, or between back-end and front-end of an application
  - Web-services are widely used today, with complex data fetched to the front-end and displayed using a mobile app or JavaScript
- JSON and XML are widely used semi-structured data models

Features of Semi-Structured Data Models

- **Flexible schema**
  - **Wide column representation: allow each tuple to have a different set of attributes, can add new attributes at any time**
  - **Sparse column representation: schema has a fixed but large set of attributes, by each tuple may store only a subset**
- **Multivalued data types**
  - **Sets, multisets**
    - **E.g.: set of interests {'basketball, 'La Liga', 'cooking', 'anime', 'jazz'}**
  - **Key-value map (or just map for short)**
    - **Store a set of key-value pairs**
    - **E.g. {(brand, Apple), (ID, MacBook Air), (size, 13), (color, silver)}**
    - **Operations on maps: *put*(key, value), *get*(key), *delete*(key)**
  - **, Arrays**
    - **Widely used for scientific and monitoring applications**

Features of Semi-Structured Data Models

- **Arrays**
  - **Widely used for scientific and monitoring applications**
  - **E.g. readings taken at regular intervals can be represented as array of values instead of (time, value) pairs**
    - **[5, 8, 9, 11] instead of {(1,5), (2, 8), (3, 9), (4, 11)}**
- **Multi-valued attribute types**
  - **Modeled using *non first-normal-form* (*NFNF*) data model**
  - **Supported by most database systems today**
- **Array database: a database that provides specialized support for arrays**
  - **E.g. compressed storage, query language extensions etc**
  - **Oracle GeoRaster, PostGIS, SciDB, etc**

Nested Data Types

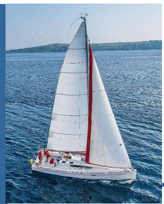
- Hierarchical data is common in many applications
- JSON: JavaScript Object Notation
  - Widely used today
- XML: Extensible Markup Language
  - Earlier generation notation, still used extensively

JSON

- Textual representation widely used for data exchange
- Example of JSON data

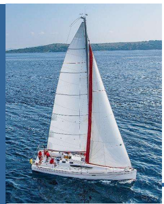
```
{
        "ID": "22222",
        "name": {
                "firstname”: "Albert",
                "lastname”: "Einstein"
        },
        "deptname": "Physics",
        "children": [
                {"firstname": "Hans", "lastname": "Einstein" },
                {"firstname": "Eduard", "lastname": "Einstein" }
        ]
}
```

- Types: integer, real, string, and
  - *Objects: are* key-value maps, i.e. sets of (attribute name, value) pairs
  - Arrays are also key-value maps (from offset to value)



JSON

- JSON is ubiquitous in data exchange today
  - Widely used for web services
  - Most modern applications are architected around on web services
- SQL extensions for
  - JSON types for storing JSON data
  - Extracting data from JSON objects using path expressions
    - E.g. *V-> ID*, or *v.ID*
  - Generating JSON from relational data
    - E.g. json.build_object('ID', 12345, 'name', 'Einstein')
  - Creation of JSON collections using aggregation
    - E.g. json_agg aggregate function in PostgreSQL
  - Syntax varies greatly across databases
- JSON is verbose
  - Compressed representations such as BSON (Binary JSON) used for efficient data storage

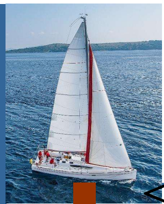


XML

- XML uses tags to mark up text
- E.g.

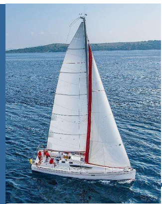
```
<course>
        <course id> CS-101 </course id>
        <title> Intro. to Computer Science </title>
        <dept name> Comp. Sci. </dept name>
        <credits> 4 </credits>
</course>
```

- Tags make the data self-documenting
- Tags can be hierarchical



Example of Data in XML

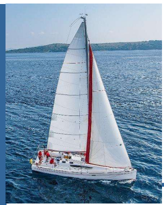
```
<purchase order>
        <identifier> P-101 </identifier>
        <purchaser>
                <name> Cray Z. Coyote </name>
                <address> Route 66, Mesa Flats, Arizona 86047, USA
        </address>
        </purchaser>
        <supplier>
                <name> Acme Supplies </name>
                <address> 1 Broadway, New York, NY, USA </address>
        </supplier>
        <itemlist>
            <item>
                <identifier> RS1 </identifier>
                <description> Atom powered rocket sled </description>
                <quantity> 2 </quantity>
                <price> 199.95 </price>
            </item>
            <item>…</item>
        </itemlist>
        <total cost> 429.85 </total cost>
      ….
</purchase order>
```



XML Cont.

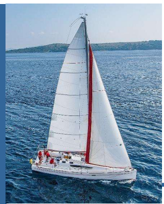
- XQuery language developed to query nested XML structures
  - Not widely used currently
- SQL extensions to support XML
  - Store XML data
  - Generate XML data from relational data
  - Extract data from XML data types
    - Path expressions
- See Chapter 30 (online) for more information

SPATIAL DATA



Spatial Data

- Spatial databases store information related to spatial locations, and support efficient storage, indexing and querying of spatial data.
  - Geographic data -- road maps, land-usage maps, topographic elevation maps, political maps showing boundaries, land-ownership maps, and so on.
    - Geographic information systems are special-purpose databases tailored for storing geographic data.
    - Round-earth coordinate system may be used
      - (Latitude, longitude, elevation)
  - Geometric data: design information about how objects are constructed . For example, designs of buildings, aircraft, layouts of integrated-circuits.
    - 2 or 3 dimensional Euclidean space with (X, Y, Z) coordinates

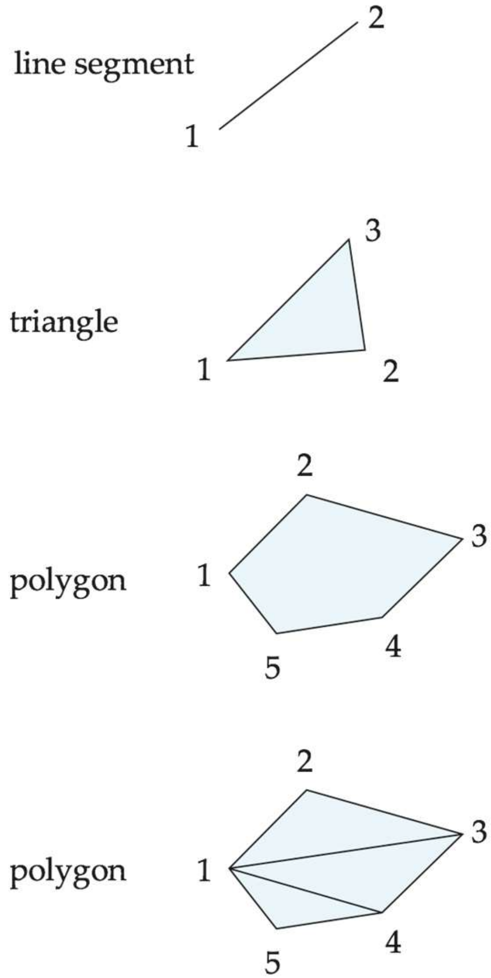


Represented of Geometric Information

Various geometric constructs can be represented in a database in a normalized fashion (see next slide)

- **A line segment can be represented by the coordinates of its endpoints.**
- **A polyline or linestring consists of a connected sequence of line segments and can be represented by a list containing the coordinates of the endpoints of the segments, in sequence.**
  - **Approximate a curve by partitioning it into a sequence of segments**
    - **Useful for two-dimensional features such as roads.**
    - **Some systems also support *circular arcs* as primitives, allowing curves to be represented as sequences of arc**
- **Polygons is represented by a list of vertices in order.**
  - **The list of vertices specifies the boundary of a polygonal region.**
  - **Can also be represented as a set of triangles (triangulation)**

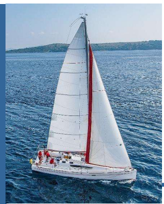
Representation of Geometric Constructs



object	representation
line segment	{(x1,y1), (x2,y2)}
triangle	{(x1,y1), (x2,y2), (x3,y3)}
polygon	{(x1,y1), (x2,y2), (x3,y3), (x4,y4), (x5,y5)}
polygon	{(x1,y1), (x2,y2), (x3,y3), ID1} {(x1,y1), (x3,y3), (x4,y4), ID1} {(x1,y1), (x4,y4), (x5,y5), ID1}

Representation of Geometric Information (Cont.)

- **Representation of points and line segment in 3-D similar to 2-D, except that points have an extra z component**
- **Represent arbitrary polyhedra by dividing them into tetrahedrons, like triangulating polygons.**
- **Alternative: List their faces, each of which is a polygon, along with an indication of which side of the face is inside the polyhedron.**
- **Geometry and geography data types supported by many databases**
  - **E.g. SQL Server and PostGIS**
  - **point, linestring, curve, polygons**
  - **Collections: multipoint, multilinestring, multicurve, multipolygon**
  - **LINESTRING(1 1, 2 3, 4 4)**
  - **POLYGON((1 1, 2 3, 4 4, 1 1))**
  - **Type conversions: *ST GeometryFromText*() and *ST GeographyFromText*()**
  - ***Operations: ST Union*(), *ST Intersection*(), …**

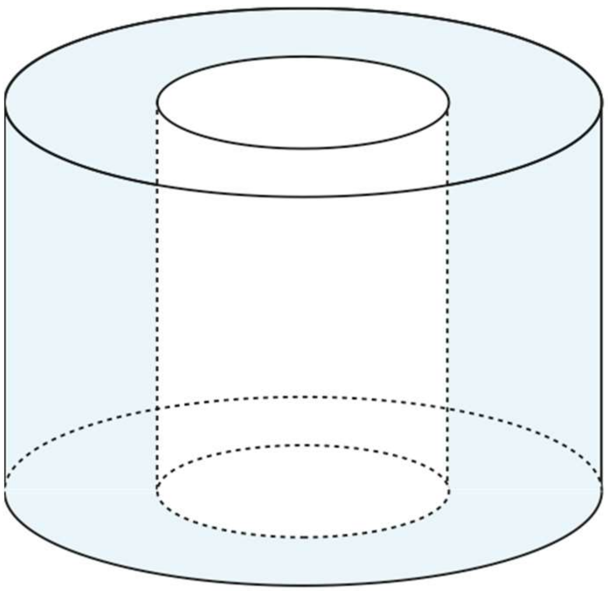


Design Databases

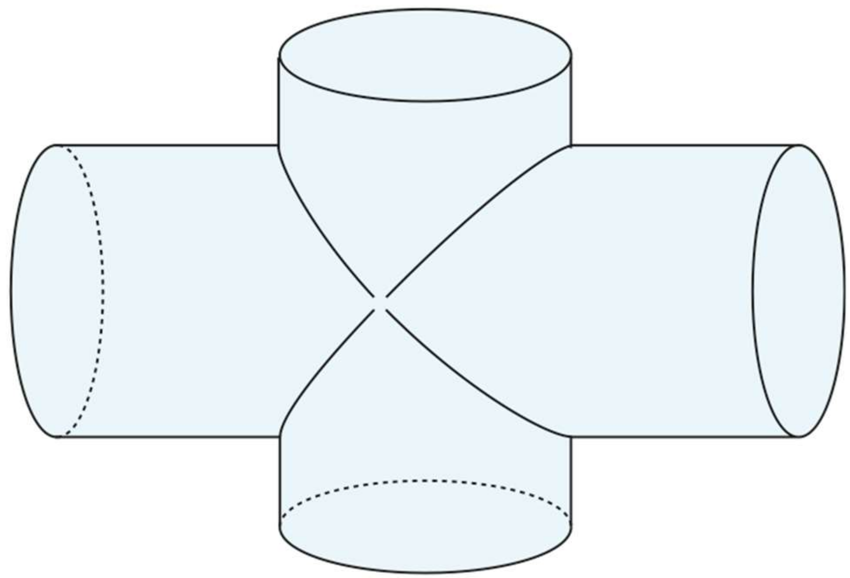
- **Represent design components as objects (generally geometric objects); the connections between the objects indicate how the design is structured.**
- **Simple two-dimensional objects: points, lines, triangles, rectangles, polygons.**
- **Complex two-dimensional objects: formed from simple objects via union, intersection, and difference operations.**
- **Complex three-dimensional objects: formed from simpler objects such as spheres, cylinders, and cuboids, by union, intersection, and difference operations.**
- **Wireframe models represent three-dimensional surfaces as a set of simpler objects.**

Representation of Geometric Constructs

- **Design databases also store non-spatial information about objects (e.g., construction material, color, etc.)**
- **Spatial integrity constraints are important.**
  - **E.g., pipes should not intersect, wires should not be too close to each other, etc.**



(a) Difference of cylinders



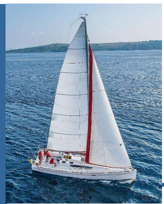
(b) Union of cylinders

Geographic Data

- **Raster data** consist of bit maps or pixel maps, in two or more dimensions.
  - Example 2-D raster image: satellite image of cloud cover, where each pixel stores the cloud visibility in a particular area.
  - Additional dimensions might include the temperature at different altitudes at different regions, or measurements taken at different points in time.
- Design databases generally do not store raster data.

Geographic Data (Cont.)

- **Vector data** are constructed from basic geometric objects: points, line segments, triangles, and other polygons in two dimensions, and cylinders, spheres, cuboids, and other polyhedrons in three dimensions.
- Vector format often used to represent map data.
  - Roads can be considered as two-dimensional and represented by lines and curves.
  - Some features, such as rivers, may be represented either as complex curves or as complex polygons, depending on whether their width is relevant.
  - Features such as regions and lakes can be depicted as polygons.



Spatial Queries

- **Region queries** deal with spatial regions. e.g., ask for objects that lie partially or fully inside a specified region
  - E.g. PostGIS *ST_Contains*(), *ST_Overlaps*(), …
- **Nearness queries** request objects that lie near a specified location.
- **Nearest neighbor queries**, given a point or an object, find the nearest object that satisfies given conditions.
- **Spatial graph queries** request information based on spatial graphs
  - E.g. shortest path between two points via a road network
- **Spatial join** of two spatial relations with the location playing the role of join attribute.
- Queries that compute intersections or **unions** of regions

پایان فصل هشتم