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>
```

Question 30-1:

a. Find the relational representation (schema) of this XML data
b. Compare XML and relational models

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

Object Orientation

- **Object-relational data model** provides richer type system
  - with complex data types and object orientation

- Applications are often written in object-oriented programming languages
  - Type system does not match relational type system
  - Switching between imperative language and SQL is troublesome

Object Orientation

- Approaches for integrating object-orientation with databases
  - Build an **object-relational database**, adding object-oriented features to a relational database Example: Oracle. PostggreSQL etc.
  - Automatically convert data between programming language model and relational model; data conversion specified by **object-relational mapping**
  - Build an **object-oriented database** that natively supports object-oriented data and direct access from programming language

Object-Relational Database Systems

a. Define a type Person with attributes ID, name and address and ID can be used as reference from other relation.

b. Create tables named people, people1. people2, etc of type Person

Ans. a

User-defined types

- **create type** *Person*
  (*ID* **varchar**(20) **primary key**,
  name **varchar**(20),
  address **varchar**(20)) **ref from**(*ID*); /* More on this later */

Ans. b

- **create table** *people* **of** *Person*;
- **create table** *people1* **of** *Person*;
- **create table** *people2* **of** *Person*;

Object-Relational Database Systems

a. Defining table type

b. User defined type can be used to define an attribute in a table.

- Table types
  - **create type** *interest* **as table** (
    topic **varchar**(20),
    degree_of_interest **int**);
  - **create table** *users* (
    ID **varchar**(20),
    name **varchar**(20),
    interests interest);
- Array, multiset data types also supported by many databases
  - Syntax varies by database

Type and Table Inheritance

Create types Student and Techer under Person

- Type inheritance
  - **create type** *Person*
    (*ID* **varchar**(20) **primary key**,
    name **varchar**(20),
    address **varchar**(20)) **ref from**(*ID*); /* More on this later */
  - **create type** *Student* **under** *Person*
    (*degree* **varchar**(20)) ;
    create type *Teacher* **under** *Person*
    (*salary* **integer**);

Type and Table Inheritance

Create a table people of type Person. Create two tables student and teacher using table inheritance of people.

Insert the following record to student and teacher tables

(201705001, “Abdullah”, ‘CSE, BUET’, ‘BSc Engineering’)

(‘T001’, ‘Sharif’, ‘CSE, BUET’, 75000)

create type *Person*
  (*ID* **varchar**(20) **primary key**,
  name **varchar**(20),
  address **varchar**(20)) **ref from**(*ID*); /* More on this later */

create table *people* **of** *Person*;

Table inheritance syntax in PostgreSQL

- **create table** *students*
  (*degree* **varchar**(20))
  inherits *people*;
  create table *teachers*
  (*salary* **integer**)
  inherits *people*;

Insert into student (ID, name, address, degree) values(201705001, “Abdullah”, ‘CSE, BUET’, ‘BSc Engineering’);

Insert into teacher (ID, name, address, salary) values (‘T001’, ‘Sharif’, ‘CSE, BUET’, 75000)

Type and Table Inheritance

Create a table people of type Person. Create two tables student and teacher using table inheritance of people.

Insert the following record to student and teacher tables

(201705001, “Abdullah”, ‘CSE, BUET’, ‘BSc Engineering’)

(‘T001’, ‘Sharif’, ‘CSE, BUET’, 75000)

Table inheritance syntax in oracle

- **create table** *people* **of** *Person*;
  create table *students* **of** *Student*
      under *people*;
  create table *teachers* **of** *Teacher*
      under *people*;

Insert into student (ID, name, address, degree) values(201705001, “Abdullah”, ‘CSE, BUET’, ‘BSc Engineering’);

Insert into teacher (ID, name, address, salary) values (‘T001’, ‘Sharif’, ‘CSE, BUET’, 75000)

Type and Table Inheritance

Question 30-2:

a. Define SQL schema for the following using type inheritance.

A car-rental company maintains a database for all vehicles in its current fleet.

For all vehicles, it includes the vehicle identification number, license number,

manufacturer, model, date of purchase, and color. Special data are included for

certain types of vehicles:

- Trucks: cargo, capacity.
- Sports cars: horsepower, renter age requirement.
- Vans: number of passengers.
- Off-road vehicles: ground clearance, drivetrain (four- or two-wheel drive).

b. insert one tuple in each table.

Reference Types

Define ID of Person as reference type. Create a type Department with attributes dept-name and head where head attribute will refer the ID of people table of type Person. Create a table department of type department.

Insert the following records into people table:

(‘T001’, ‘Sharif’, ‘CSE, BUET’, 75000)

(‘T002’, ‘Rafique’, ‘EEE, BUET’, 75000)

(‘T003’, ‘Atique’, ‘CSE, BUET’, 75000)

Insert a records into department table such that ‘Atique’ is the head of CSE department using ID as reference type (User defined reference).

- Creating reference types
  - **create type** *Person*
    (*ID* **varchar**(20) **primary key**,
    name **varchar**(20),
    address **varchar**(20)
  - salary int)
    ref from(*ID*);

  User defined reference

  insert into *departments* **values** (‘CSE', ‘T003’)

create table *people* **of** *Person*;
create type *Department* (
dept_name **varchar(20)**,
head **ref**(*Person*) **scope** *people*);
create table *departments* **of** *Department*

Reference Types

Define ID of Person as reference type. Create a type Department where head attribute will refer the ID of people table of type Person. Create a table department of type department.

Insert the following records into people table:

(‘T001’, ‘Sharif’, ‘CSE, BUET’, 75000)

(‘T002’, ‘Rafique’, ‘EEE, BUET’, 75000)

(‘T003’, ‘Atique’, ‘CSE, BUET’, 75000)

Insert a records into department table such that ‘Atique’ is the head of CSE department using ID as reference type using system generated references .

Insert into department values (‘CSE’, null)

System generated references can be retrieved using subqueries

(**select ref**(*p*) **from** *people* **as** *p* **where** *ID* = ‘T003')

Update the reference of head, CSE by system generated reference.

Update department
Set head = (**select ref**(*p*) **from** *people* **as** *p* **where** *ID* = ‘T003’)
Where dept_name = ‘CSE’

Reference Types

Query: Find the department name, name of the head and address from the department table.

```
create table people of Person;
create type Department (
    dept_name varchar(20),
    head ref(Person) scope people);
create table departments of
Department
```

```
create type Person
   (ID varchar(20) primary key,
    name varchar(20),
    address varchar(20))
    ref from(ID);
```

Using references in **path expressions**

- **select** dept_name**,** *head->name*, *head->address*
  from *departments*;

Question 31-1: Write the output of the above SQL expression

Reference Types

Question 31-2:

Create type customer with attributes c-ID, name, street and city. Use reference type for c-ID. Insert the following records into customer table.

(‘C001’, ‘Arif’, ‘North’, ‘Dhaka’)

(‘C002’, ‘Abdullah’, ‘South’, ‘Dhaka’)

Create type account with attributes Acc-ID, type, owner where owner is reference type and refers to customer table.

Insert an account with Acc-ID =A0001 and type = ‘savings’. The owner of the account is Abdullah.

Implement the above using

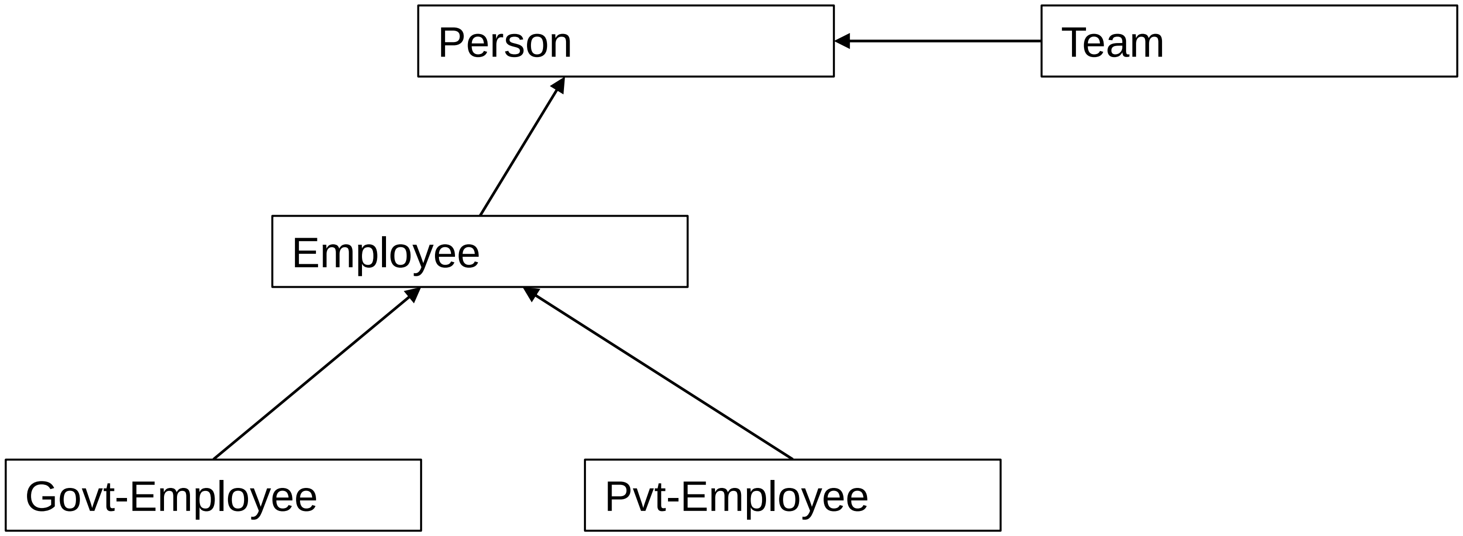
a. c-ID as reference
b. System generated reference
c. Compare the performance of the above implementation with the same with foreign key reference.

Practice problem (No need to submit in CP)

A person has NID, name, street, city, thana, district and age. A person may be an employee with special attributes as salary and qualification (highest degree). An employee may be government or private. For government employees, special attributes are ministry and designation. For non-government employees, special attributes are company name and position.

National team is formed with the attributes as id, team leader, game, organizing country, and date. The attribute team leader is reference type that refers to person. You have to define NID as reference.

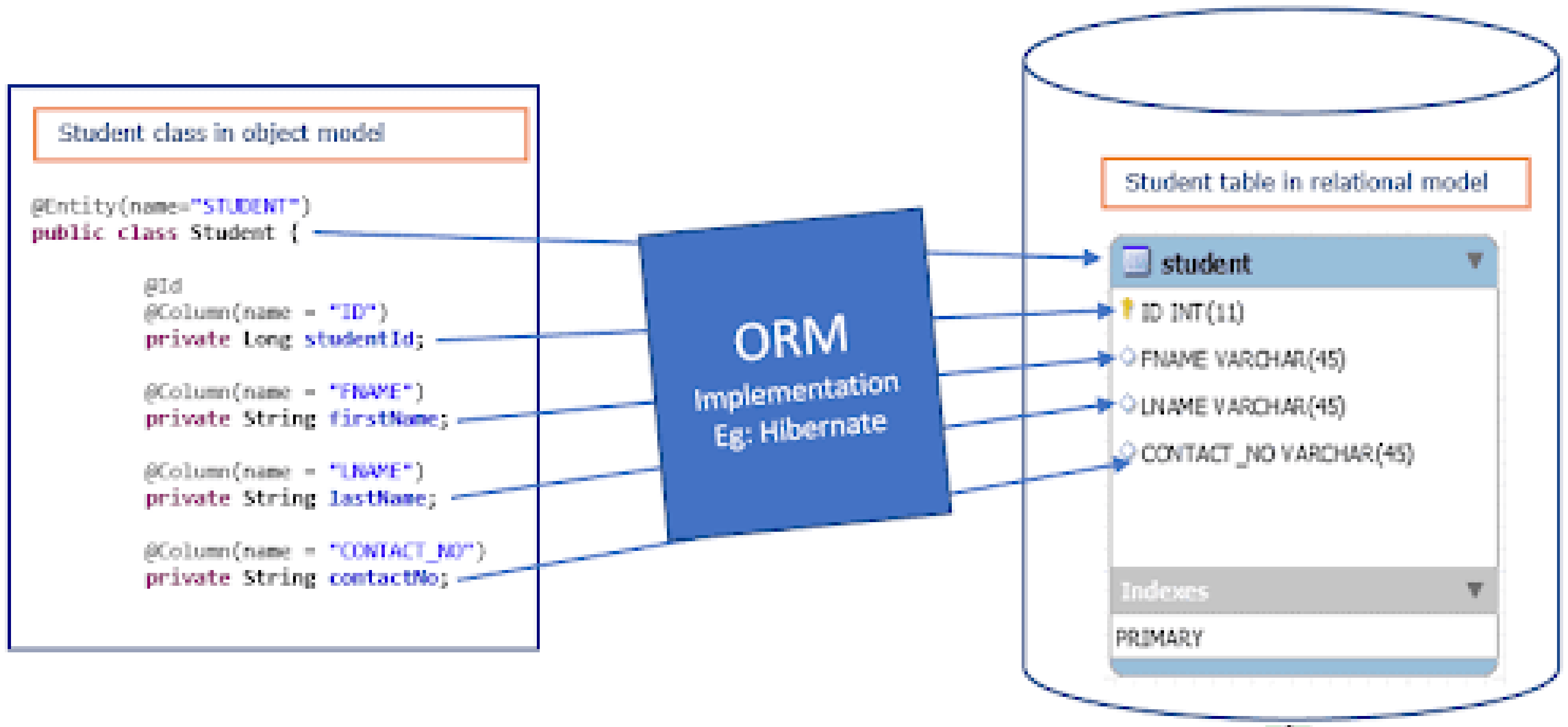
a. Define types and inheritance
b. Create table for each type with table inheritance using PostgreSQL and Oracle formats and insert two tuples in each table.
c. Define reference type for national team and appoint Abdullah with NID 1234567890 as team leader using system generated reference. Insert two tuples in national team relation.
d. Write path expression query to
    i. Find NID and name of all employees with salary >50000.
    ii. Find name, street and city of the team leader with id = ‘T005’


Person
Team
Employee
Govt-Employee
Pvt-Employee

Object-Relational Mapping

- Object-relational mapping (ORM) systems allow
  - Specification of mapping between programming language objects and database tuples
  - Automatic creation of database tuples upon creation of objects
  - Automatic update/delete of database tuples when objects are update/deleted
  - Interface to retrieve objects satisfying specified conditions
    - Tuples in database are queried, and object created from the tuples

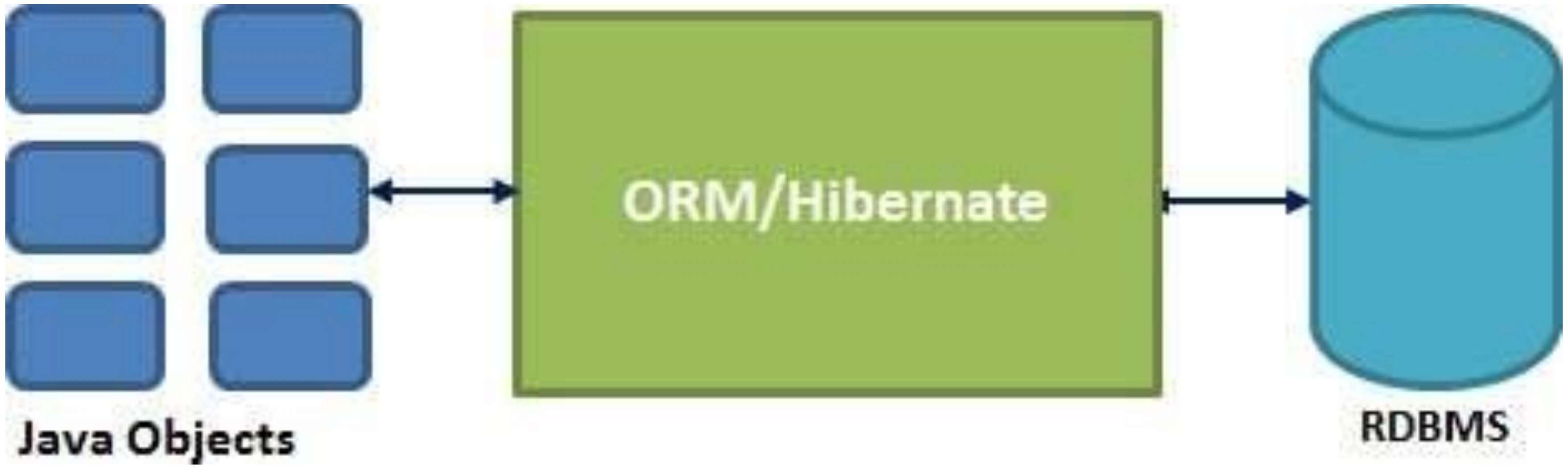
Object-Relational Mapping



```
@Entity(name="STUDENT")
public class Student {

        @Id
        @Column(name = "ID")
        private Long studentId;

        @Column(name = "FNAME")
        private String firstName;

        @Column(name = "LNAME")
        private String lastName;

        @Column(name = "CONTACT_NO")
        private String contactNo;
```

ORM implements responsibility of mapping the Object to Relational Model. JavaByDeveloper

Object-Relational Mapping (Hibernate)



Object-Relational Mapping

Sr. No.	Advantages
1	Let’s business code access objects rather than DB tables.
2	Hides details of SQL queries from OO logic.
3	Based on JDBC 'under the hood.'
4	No need to deal with the database implementation.
5	Entities based on business concepts rather than database structure.
6	Transaction management and automatic key generation.
7	Fast development of application.

Object-Relational Mapping Solution

An ORM solution consists of the following four entities −

Sr. No.	Solutions
1	An API to perform basic CRUD operations on objects of persistent classes.
2	A language or API to specify queries that refer to classes and properties of classes.
3	A configurable facility for specifying mapping metadata.
4	A technique to interact with transactional objects to perform dirty checking, lazy association fetching, and other optimization functions.

Object-Relational Mapping Solution

Java ORM Frameworks

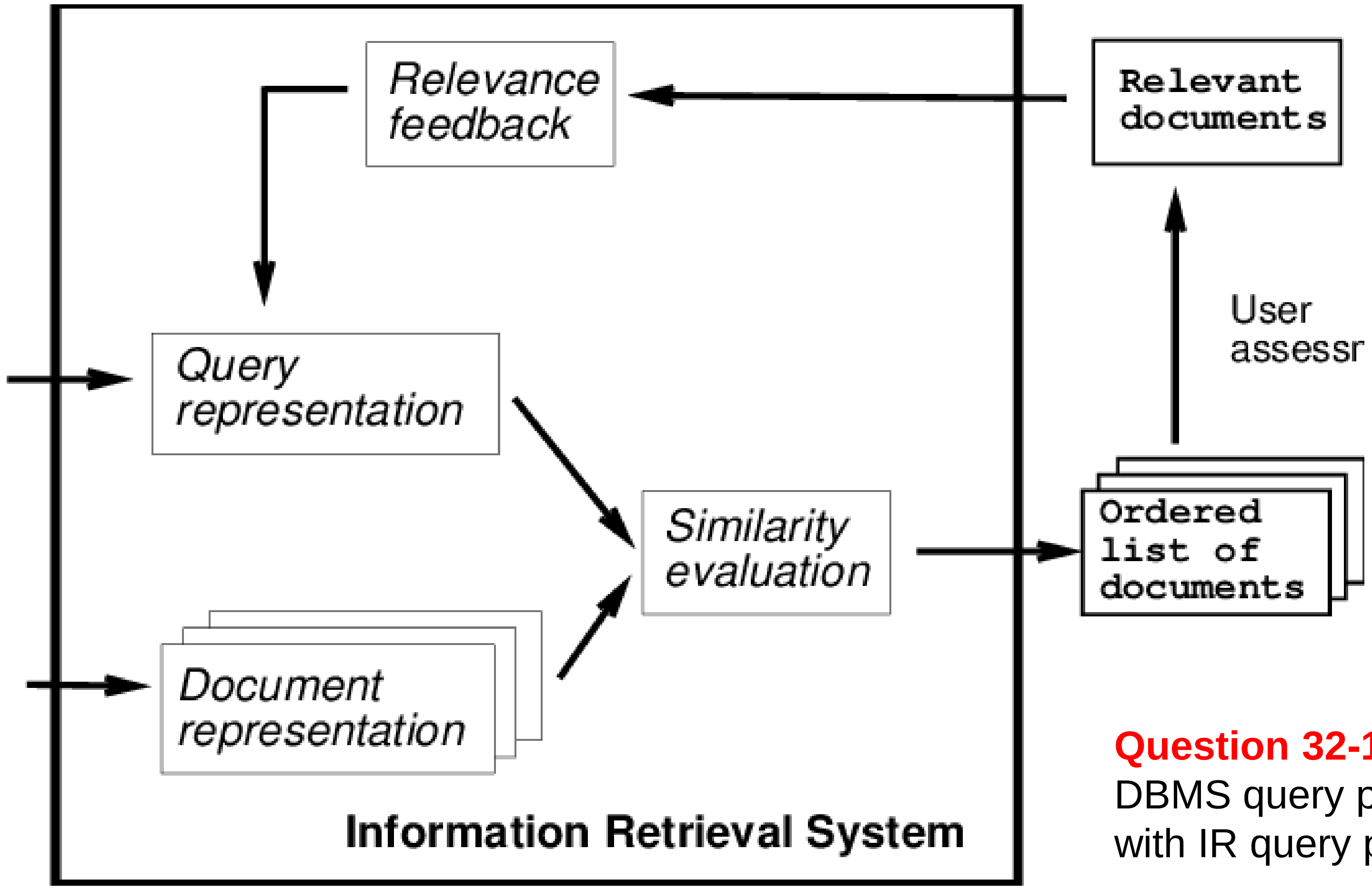
There are several persistent frameworks and ORM options in Java. A persistent framework is an ORM service that stores and retrieves objects into a relational database.

•Enterprise JavaBeans Entity Beans
•Java Data Objects
•Castor
•TopLink
•Spring DAO
•Hibernate

Textual Data (Information Retrieval Systems)

- **Information retrieval**: querying of unstructured data
  - Simple model of keyword queries: given query keywords, retrieve documents containing all the keywords
  - More advanced models rank relevance of documents
  - Today, keyword queries return many types of information as answers
    - E.g., a query “cricket” typically returns information about ongoing cricket matches
- Relevance ranking
  - Essential since there are usually many documents matching keywords

Information Retrieval System



Question 32-1: Compare DBMS query processing with IR query processing

Ranking using TF-IDF

- Term: keyword occurring in a document/query
- **Term Frequency:** *TF*(*d*, *t*), the relevance of a term *t* to a document *d*
  - One definition*:* *TF*(*d*, *t*) = *log*(1 + *n*(*d,t*)/*n*(*d*))
    where
    - *n*(*d,t*) = number of occurrences of term *t* in document *d*
    - *n*(*d*) = number of terms in document *d*

***Question 32-2:** A document d is given as follows:*

A person has NID, name, street, city, thana, district and age. A person may be an employee with special attributes as salary and qualification (highest degree). An employee may be government or private. For government employees, special attributes are ministry and designation. For non-government employees, special attributes are company name and position.

National team is formed with the attributes as id, team leader, game, organizing country, and date. The attribute team leader is reference type that refers to person. You have to define NID as reference.

a. *Find TF (d, attribute)*

b. *Why logarithmic function has been considered instead of n(d,t)?*

Ranking using TF-IDF

- **Inverse document frequency**: IDF(t)
  - One definition: $IDF(t) = 1/n(t)$
- where n(t) denotes the number of documents (among those indexed by the system) that contain the term t.

Question 32-3: *A document d1 is given as follows:*

A person has NID, name, street, city, thana, district and age. A person may be an employee with special attributes as salary and qualification (highest degree). An employee may be government or private. For government employees, special attributes are ministry and designation. For non-government employees, special attributes are company name and position. *TF (d1, attribute)* = *log*(1 + *2/50*)

Another document d2 is given as follows:
Define SQL schema for the following using type inheritance.
A car-rental company maintains a database for all vehicles in its current fleet. For all vehicles, it includes the vehicle identification number, license number, manufacturer, model, date of purchase, and color.

a. *Find IDF(attribute)*

b. *Find IDF(for)*

Ranking using TF-IDF

- **Relevance** of a document *d* to a set of terms *Q*
  - *One definition:* $r(d, Q) = \sum_{t \in Q} TF(d, t) * IDF(t)$
  - Other definitions
    - take **proximity** of words into account
    - **Stop words** are often ignored

Question 32-4: *A document d1 is given as follows:*

A person has NID, name, street, city, thana, district and age. A person may be an employee with special attributes as salary and qualification (highest degree). An employee may be government or private. For government employees, special attributes are ministry and designation. For non-government employees, special attributes are company name and position.

Another document d2 is given as follows:
An employee may be government or private. For government employees, special attributes are ministry and designation. For non-government employees, special attributes are company name and position.

a. Find relevance r (d1, {attribute, salary})

b. Find relevance r (d2, {attribute, salary})

c. Find ranking of the query {attribute, salary}

Ranking using TF-IDF

- **Relevance** of a document *d* to a set of terms *Q*
  - *One definition:* $r(d, Q) = \sum_{t \in Q} TF(d, t) * IDF(t)$
  - Other definitions
    - take **proximity** of words into account
    - **Stop words** are often ignored

- **Proximity** of the terms in the document: If the terms occur close to each other in the document, the document will be ranked higher than if they occur far apart.
- The formula for r(d, Q) can be modified to take proximity of the terms into account

- Example: relevance (d, North South University) should be higher than the

relevance (d, North ……. South …………..University)

Ranking using TF-IDF

- **Relevance** of a document *d* to a set of terms *Q*
  - *One definition:* $r(d, Q) = \sum_{t \in Q} TF(d, t) * IDF(t)$
  - Other definitions
    - take **proximity** of words into account
    - **Stop words** are often ignored

- Given a query Q, the job of an information-retrieval system is to return documents in descending order of their relevance to Q.

- Since there may be a very large number of documents that are relevant, information-retrieval systems typically return only the first few documents with the highest degree of estimated relevance and permit users to interactively request further documents.

- Example: Google search engine query result

Ranking Using Hyperlinks

- Hyperlinks provide very important clues to importance
- Google introduced PageRank, a measure of popularity/importance based on hyperlinks to pages
  - Pages hyperlinked from many pages should have higher PageRank
  - Pages hyperlinked from pages with higher PageRank should have higher PageRank
  - Formalized by **random walk** model

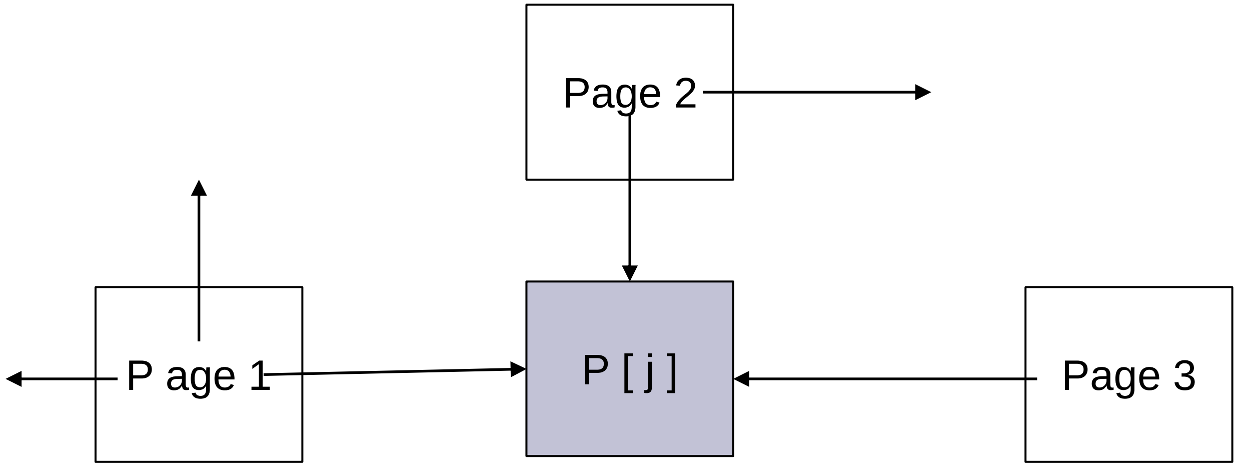
Ranking Using Hyperlinks

- Let $T[i, j]$ be the probability that a random walker who is on page *i* will click on the link to page *j*
  - Assuming all links are equal, $T[i, j] = 1/N_i$, where Ni is the number of links out of page i.
- Then PageRank[j] for each page j can be defined as
  - $P[j] = \delta/N + (1 - \delta) * \sum_{i=1}^{N} (T[i, j] * P[i])$

Where N = total number of pages, and δ a constant usually set to 0.15

Ranking Using Hyperlinks

- Let *T*[*i*, *j*] be the probability that a random walker who is on page *i* will click on the link to page *j*
  - Assuming all links are equal, *T*[*i*, *j*] = 1 ⁄ *Ni* , where Ni is the number
  - of links out of page i.
- Then PageRank[j] for each page j can be defined as
  - $P[j] = \delta/N + (1 - \delta) * \sum_{i=1}^{N} (T[i, j] * P[i])$
  - Where *N* = total number of pages, and δ a constant usually set to 0.15



Ranking Using Hyperlinks

- Definition of PageRank is circular, but can be solved as a set of linear equations
  - Simple iterative technique works well
  - Initialize all P[i] = 1/N
  - In each iteration use equation $P[j] = \delta/N + (1 - \delta) * \sum_{i=1}^{N} (T[i, j] * P[i])$ to update P
  - Stop iteration when changes are small, or some limit (say 30 iterations) is reached.

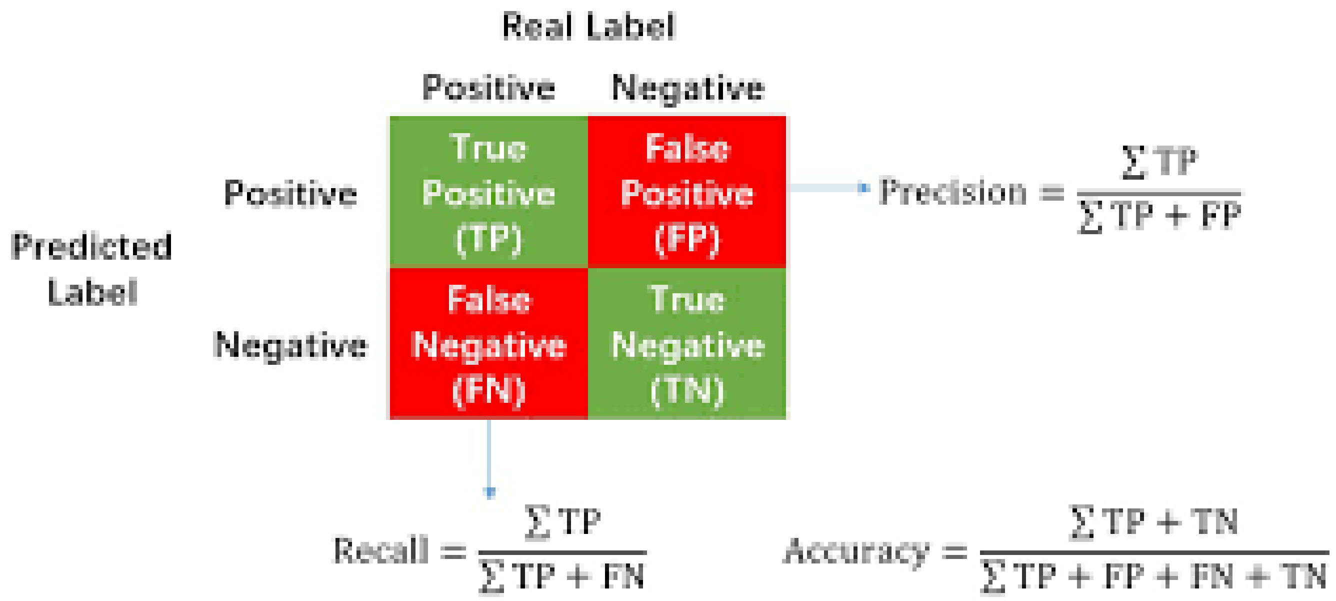
- Other measures of relevance are also important. For example:
  - Keywords in anchor text
  - Number of times who ask a query click on a link if it is returned as an answer

Retrieval Effectiveness

- Measures of effectiveness
  - **Precision**: what percentage of returned results are actually relevant
  - **Recall**: what percentage of relevant results were returned
  - At some number of answers, e.g. precision@10, recall@10

Retrieval Effectiveness

- **Precision**: what percentage of returned results are actually relevant
- **Recall**: what percentage of relevant results were returned
- At some number of answers, e.g. precision@10, recall@10

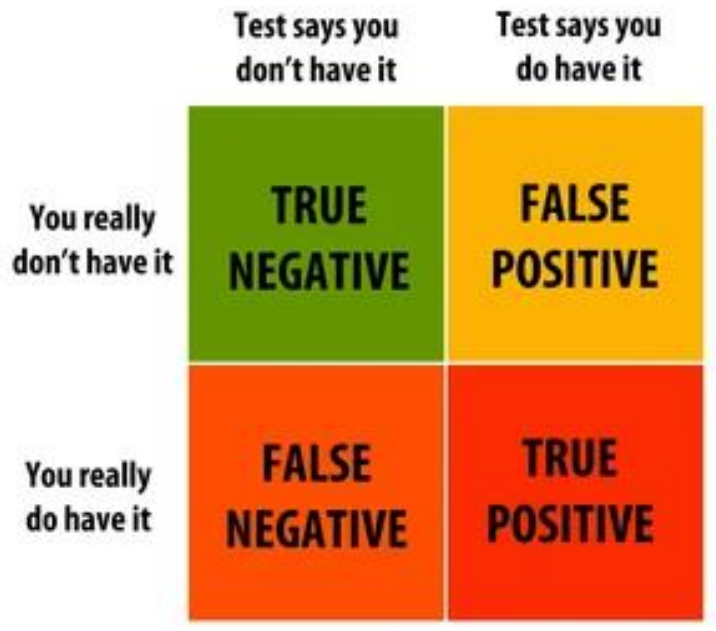


$$\text{Precision} = \frac{\sum TP}{\sum TP + FP}$$

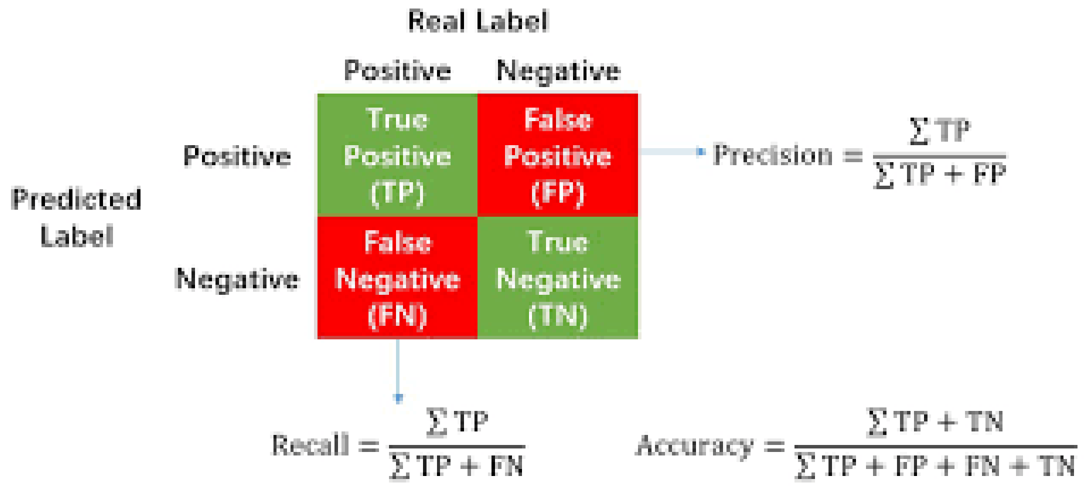
$$\text{Recall} = \frac{\sum TP}{\sum TP + FN}$$

$$\text{Accuracy} = \frac{\sum TP + TN}{\sum TP + FP + FN + TN}$$

Retrieval Effectiveness



shutterstock.com · 1506772313

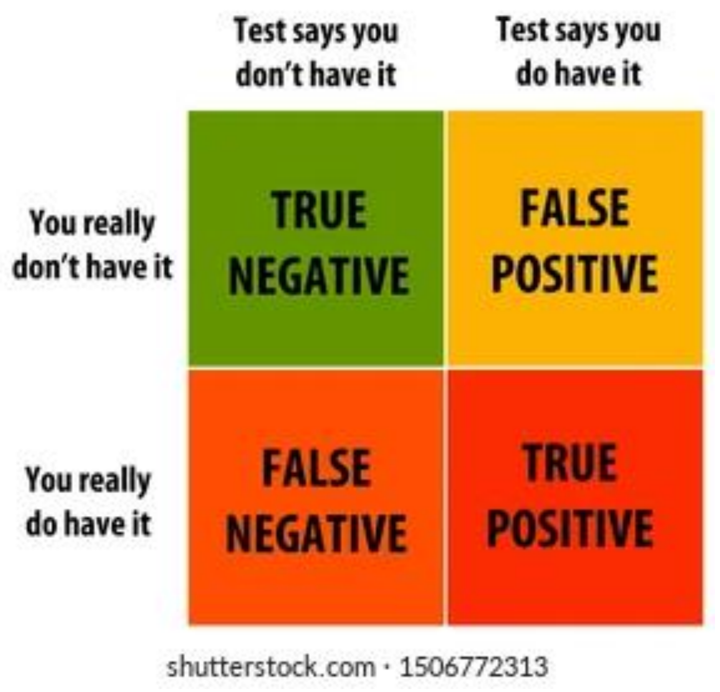


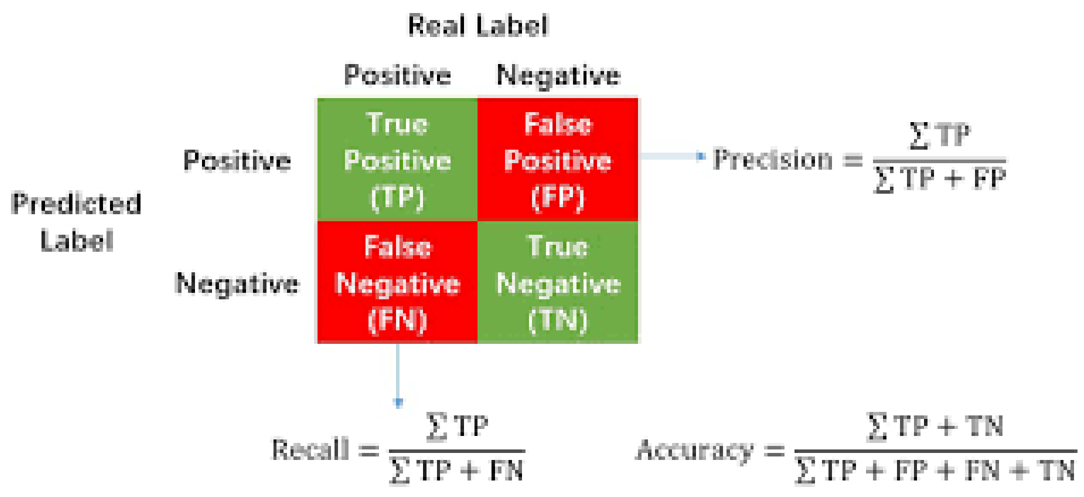
$$\text{Precision} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FN}}$$

$$\text{Accuracy} = \frac{\sum \text{TP} + \text{TN}}{\sum \text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Retrieval Effectiveness





$$\text{Precision} = \frac{\sum TP}{\sum TP + FP}$$

$$\text{Recall} = \frac{\sum TP}{\sum TP + FN}$$

$$\text{Accuracy} = \frac{\sum TP + TN}{\sum TP + FP + FN + TN}$$

Question 33-1: Compare database query processing and information retrieval query processing

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)

Spatial Data

- **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

- 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.

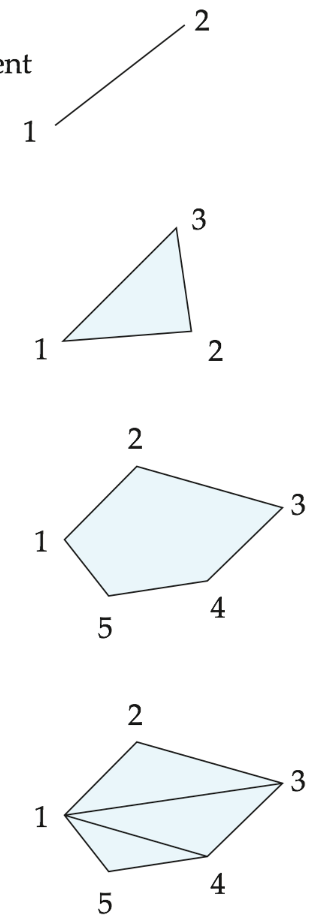
    - Approximate a curve by partitioning it into a sequence of segments
        - Useful for two-dimensional features such as roads.
        - Some systems also support *circular arcs* as primitives, allowing curves to be represented as sequences of arc

Represented of Geometric Information

Various geometric constructs can be represented in a database in a normalized fashion

- **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}

Question 33-2: A triangle has the coordinates (0, 0), (5, 0), (0, 5). Represent the triangle by

a. Lines
b. Points

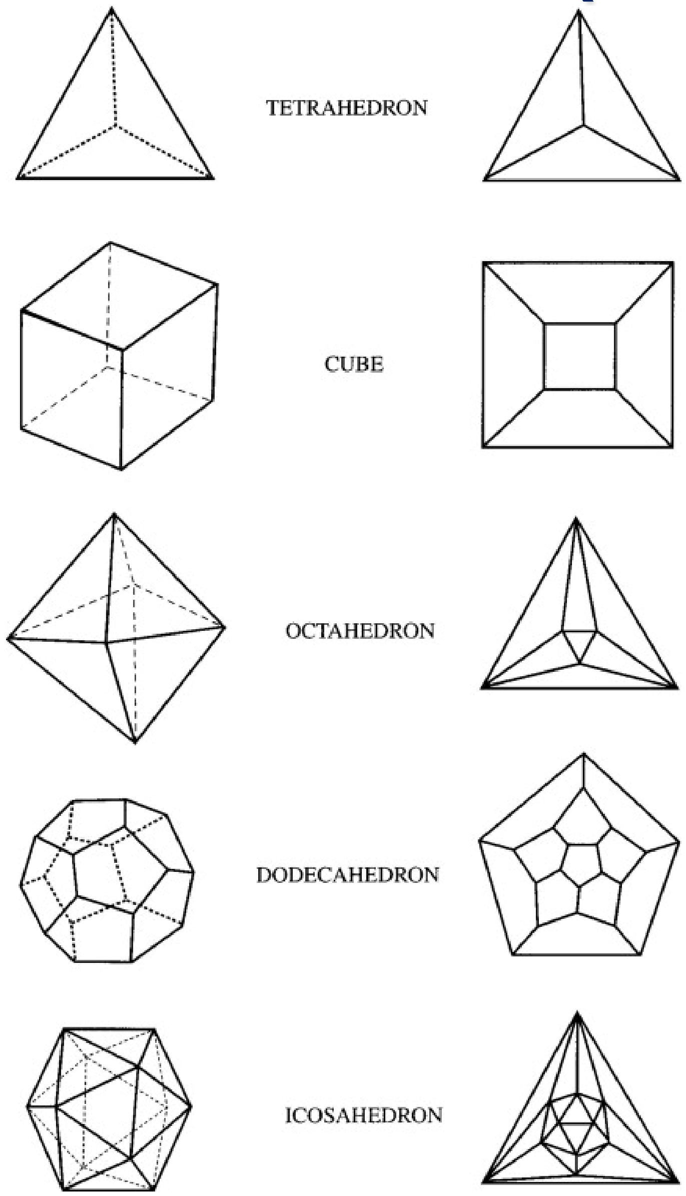
in the database.

Question 33-3: Show the representation of the given polygon using a single tuple in relational model.

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.

Question 34-1: Explain how the polyhedral can be represented by tetrahedrs



Representation of Geometric Information (Cont.)

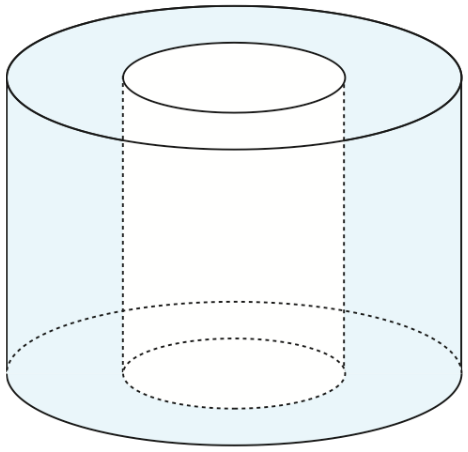
- Geometry and geography data types supported by many databases
  - E.g. SQL Server and PostGIS
  - point, linestring, curve, polygons
  - Collections: multipoint, multilinestring, multicurve, multipolygon
  - LINESTRING(1 1, 2 3, 4 4)
  - POLYGON((1 1, 2 3, 4 4, 1 1))
  - Type conversions: *ST GeometryFromText*() and *ST GeographyFromText*()
  - *Operations: ST Union*(), *ST Intersection*(), …

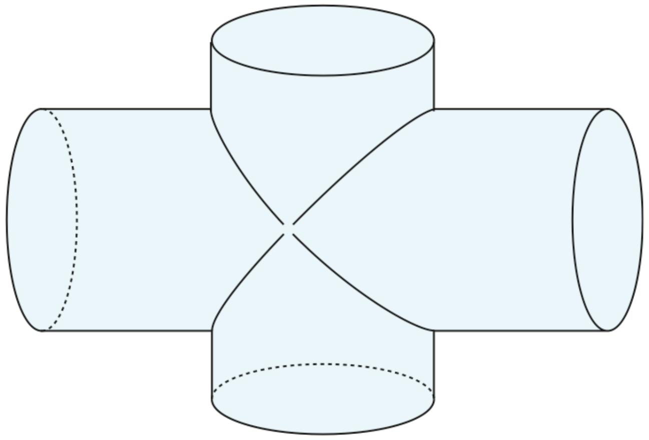
Design Databases

- 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.
- Raster data are often represented as tiles, each covering a fixed-size area. A larger area can be displayed by displaying all the tiles that overlap with the area.
  - 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.)

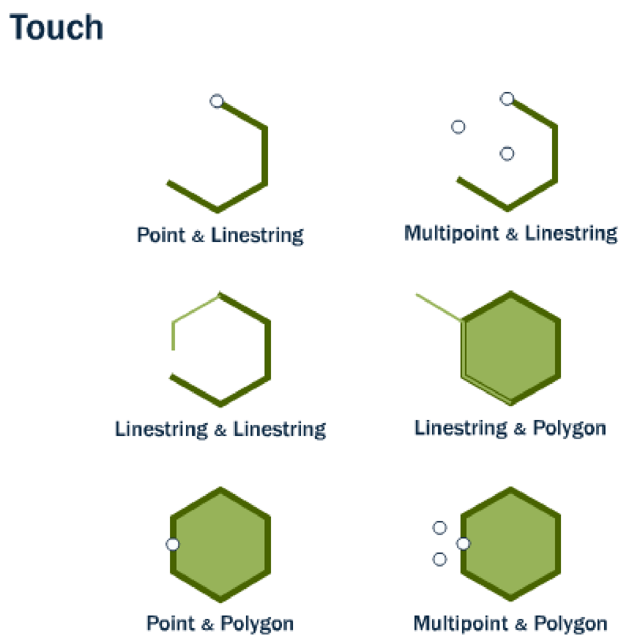
- **Vector data** are constructed from basic geometric objects: points, line segments, triangles, and other polygons in two dimensions, and cylinders, spheres, cuboids, and other polyhedrons in three dimensions.

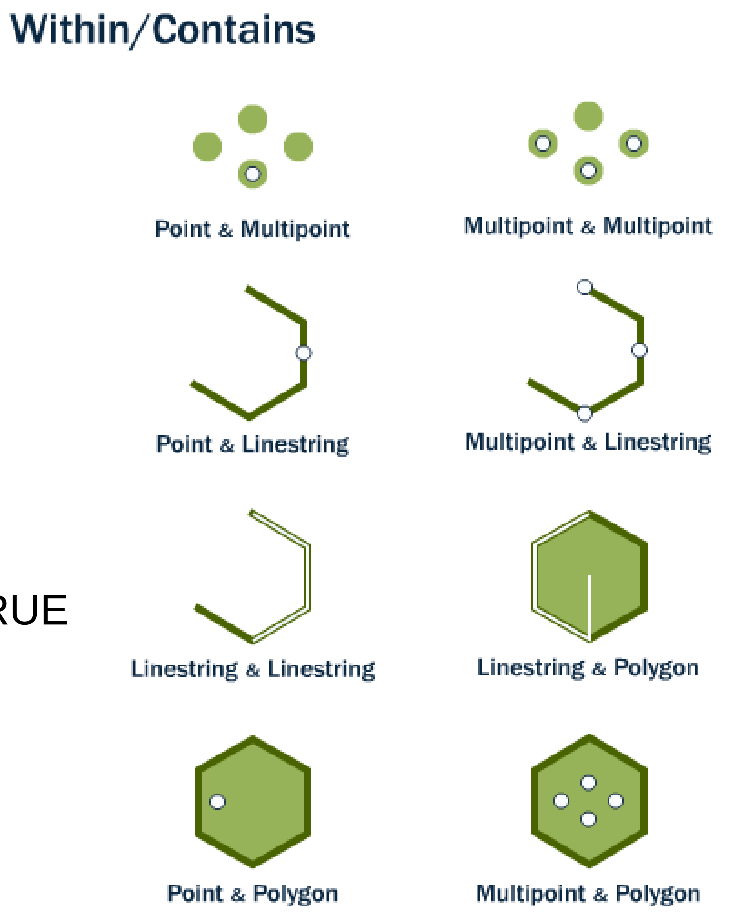
- Vector format often used to represent map data.
  - Roads can be considered as two-dimensional and represented by lines and curves.
  - Some features, such as rivers, may be represented either as complex curves or as complex polygons, depending on whether their width is relevant.
  - Features such 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*(), *ST Disjoint*(), *ST Touches*(). …
- Suppose we have a *shop* relation, with an attribute *location* of type *point*, and a geography object of type *polygon*.
- Then the *ST Contains*() function can be used to retrieve all shops whose location is contained in the given polygon.

Spatial Queries

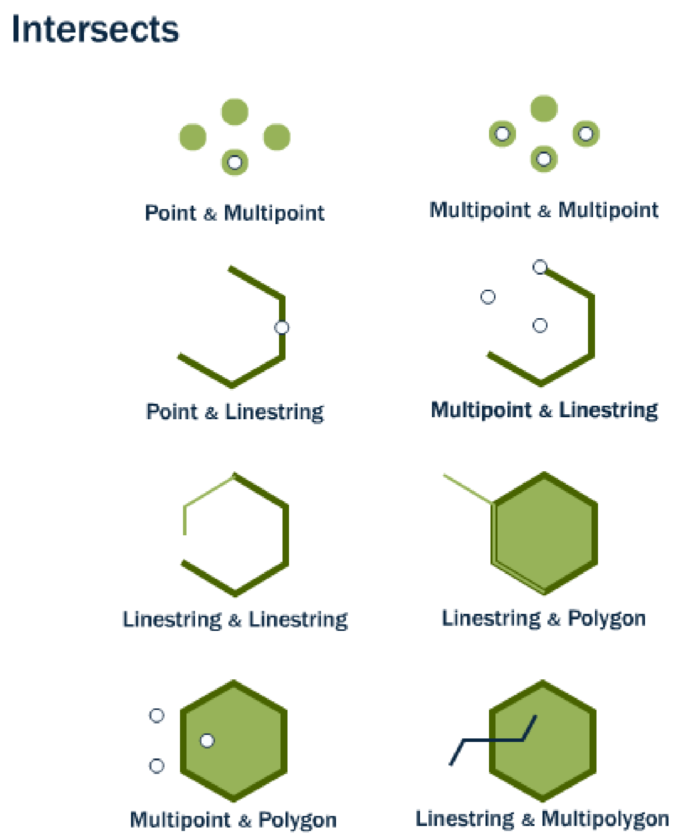


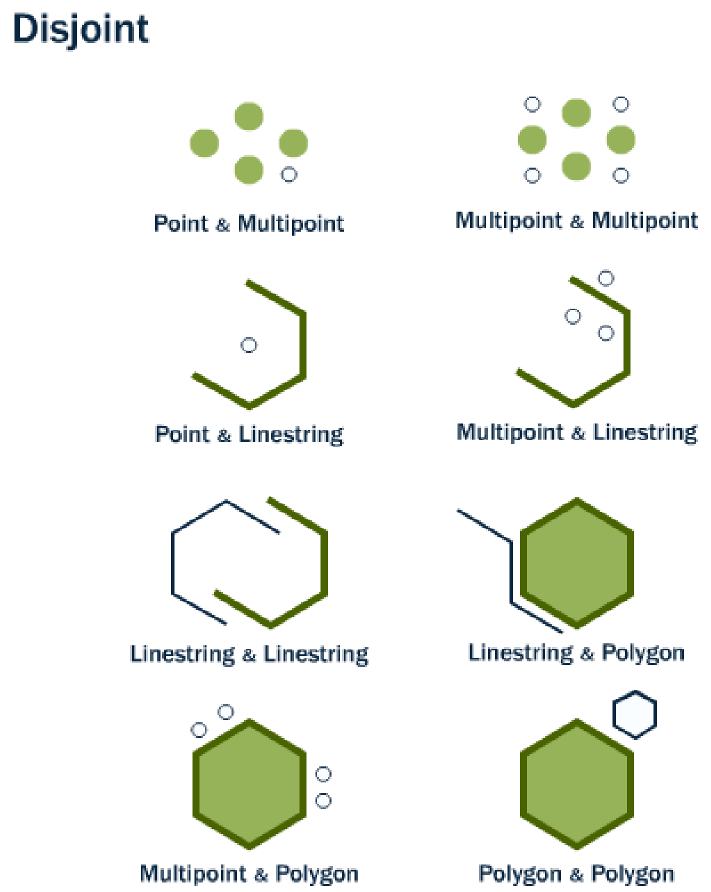


ST_Touches(geometry A, geometry B) returns TRUE if either of the geometries' boundaries intersect or if only one of the geometry's interiors intersects the other's boundary.

ST_Contains(geometry A, geometry B) returns TRUE if the second geometry is completely contained by the first geometry.

Spatial Queries





ST_Intersects(geometry A, geometry B) returns t (TRUE) if the two shapes have any space in common, i.e., if their boundaries or interiors intersect.

The opposite of ST_Intersects is **ST_Disjoint(geometry A , geometry B)**.

Spatial Queries

- **Nearness queries** request objects that lie near a specified location.
- A query to find all restaurants that lie within a given distance of a given point is an example of a nearness query.
- **Nearest neighbor queries**, given a point or an object, find the nearest object that satisfies given conditions.
- we may want to find the nearest gasoline station
- The PostGIS *ST Distance*() function gives the minimum distance between two such objects, and can be used to find objects that are within a specified distance from a point or region. Nearest neighbors can be found by finding objects with minimum distance.

Spatial Queries

- **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

```
SELECT
  subways.name AS subway_name,
  neighborhoods.name AS neighborhood_name,
  neighborhoods.boroname AS borough
FROM nyc_neighborhoods AS neighborhoods
JOIN nyc_subway_stations AS subways
ON ST_Contains(neighborhoods.geom, subways.geom)
WHERE subways.name = 'Broad St';
```