Complex Data Types

Lê Hồng Hải
UET-VNUH

Overview

1. Semi-Structured
2. JSON
3. XML
4. Spatial Data

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

Semi-Structured Data

- 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

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 as an XML Alternative

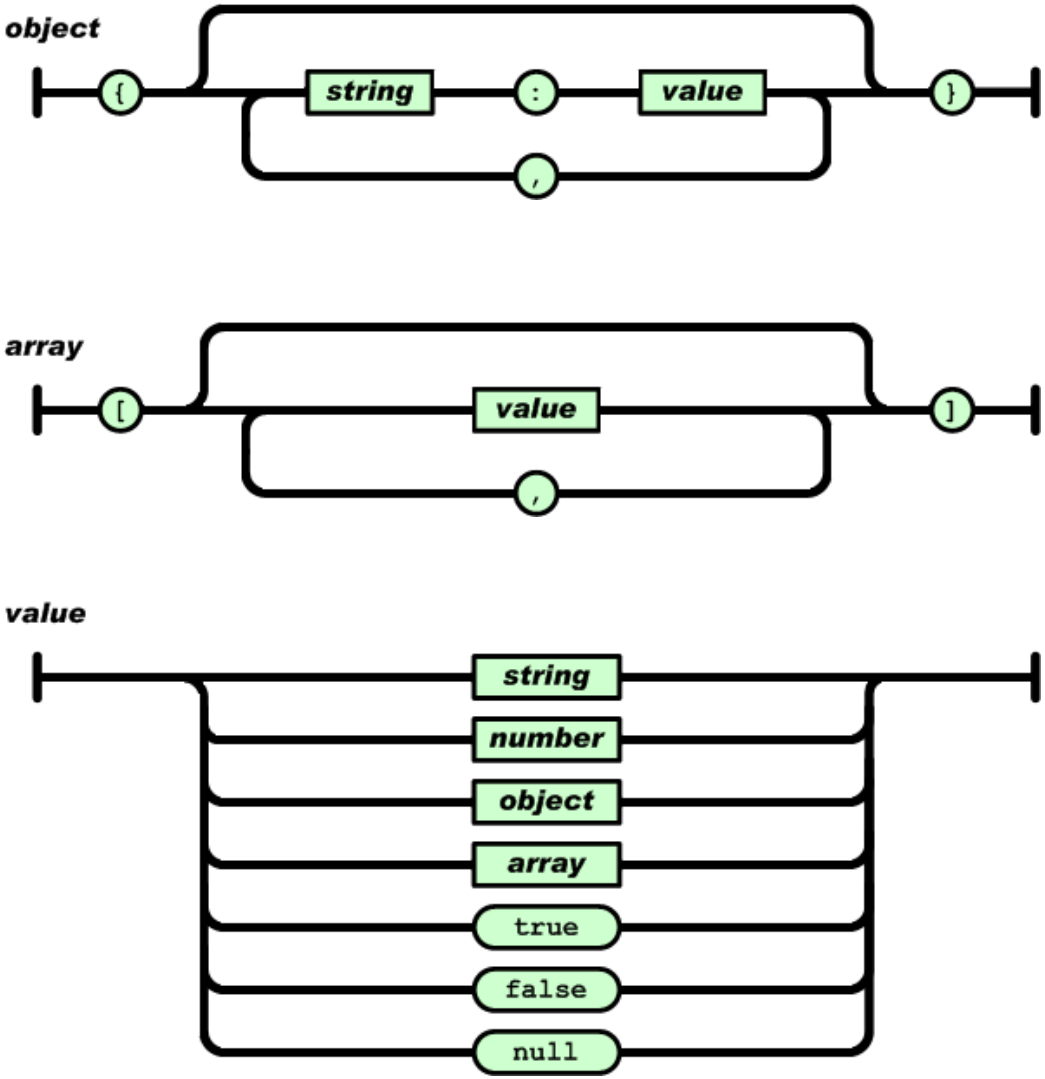
- JSON is a light-weight alternative to XML for data-interchange
- JSON = JavaScript Object Notation
  - It's really language independent
  - most programming languages can easily read it and instantiate objects or some other data structure
- Defined in RFC 4627
- Started gaining tracking ~2006 and now widely used

Example

```
{"firstName": "John",
 "lastName" : "Smith",
 "age"        : 25,
 "address"   :
   {"streetAdr" : "21 2nd Street",
    "city"       : "New York",
    "state"      : "NY",
    "zip"        : "10021"},
 "phoneNumber":
   [{"type"  : "home",
    "number": "212 555-1234"},
    {"type"  : "fax",
    "number" : "646 555-4567"}]
}
```

- This is a JSON object with five key-value pairs
- Objects are wrapped by curly braces
- There are no object IDs
- Keys are strings
- Values are numbers, strings, objects or arrays
- Arrays are wrapped by square brackets

The BNF is simple



Evaluation

- JSON is simpler than XML and more compact
  - No closing tags, but if you compress XML and JSON the difference is not so great
  - XML parsing is hard because of its complexity
- JSON has a better fit for OO systems than XML
- JSON is not as extensible as XML
- Preferred for simple data exchange by many
- Less syntax, no semantics

XML Example

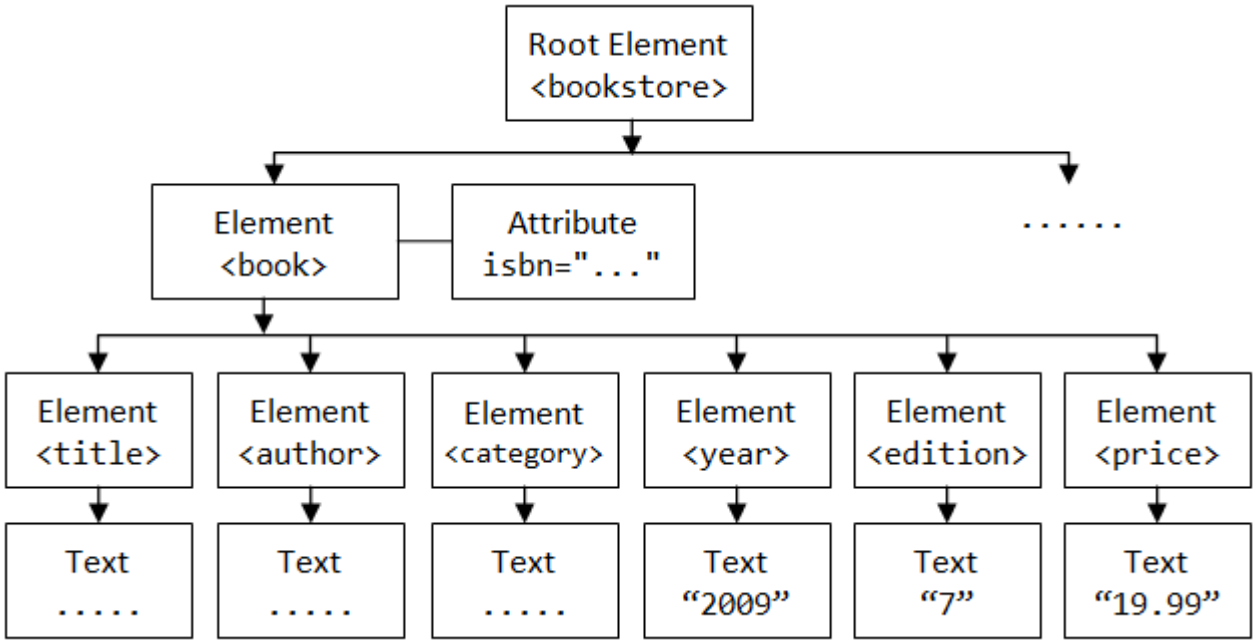
```
<!-- bookstore.xml -->
- <bookstore>
  - <book ISBN="0123456001">
      <title>Java For Dummies</title>
      <author>Tan Ah Teck</author>
      <category>Programming</category>
      <year>2009</year>
      <edition>7</edition>
      <price>19.99</price>
    </book>
  - <book ISBN="0123456002">
      <title>More Java For Dummies</title>
      <author>Tan Ah Teck</author>
      <category>Programming</category>
      <year>2008</year>
      <price>25.99</price>
    </book>
  - <book ISBN="0123456010">
      <title>The Complete Guide to Fishing</title>
      <author>Bill Jones</author>
      <author>James Cook</author>
      <author>Mary Turing</author>
      <category>Fishing</category>
      <category>Leisure</category>
      <language>French</language>
      <year>2000</year>
      <edition>2</edition>
      <price>49.99</price>
    </book>
</bookstore>
```


10

Well-Form XML Documents

- An XML document is *well-formed*, if its structure meets the XML specification
- A well-formed XML document exhibits a tree-like structure, and can be processed by an XML processor

XML Documents are Trees



Example of Data in XML

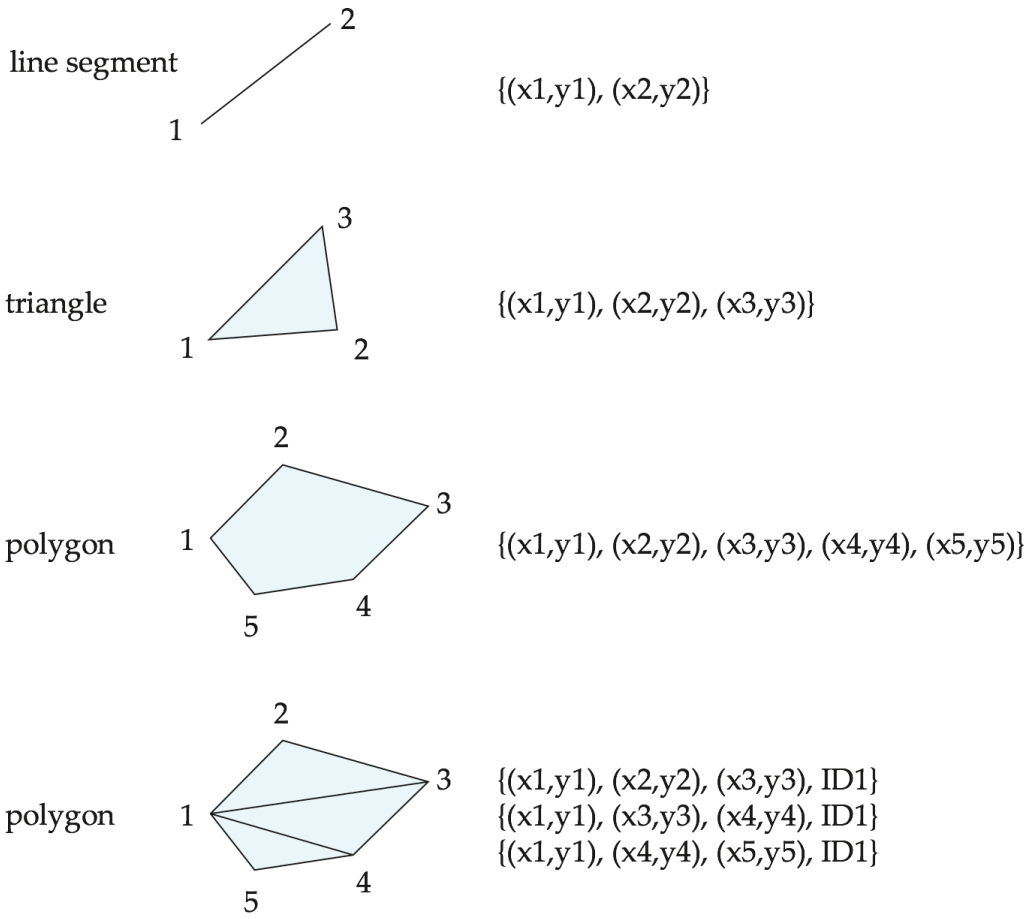
```
<purchase order>
        <identifier> P-101 </identifier>
        <purchaser>
                <name> Cray Z. Coyote </name>
                <address> Route 66, Mesa Flats, Arizona 86047, USA
                </address>
        </purchaser>
        <supplier>
                <name> Acme Supplies </name>
                <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>
```

- **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
- Approaches for integrating object-orientation with databases

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.
  - **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

Representation of Geometric Constructs

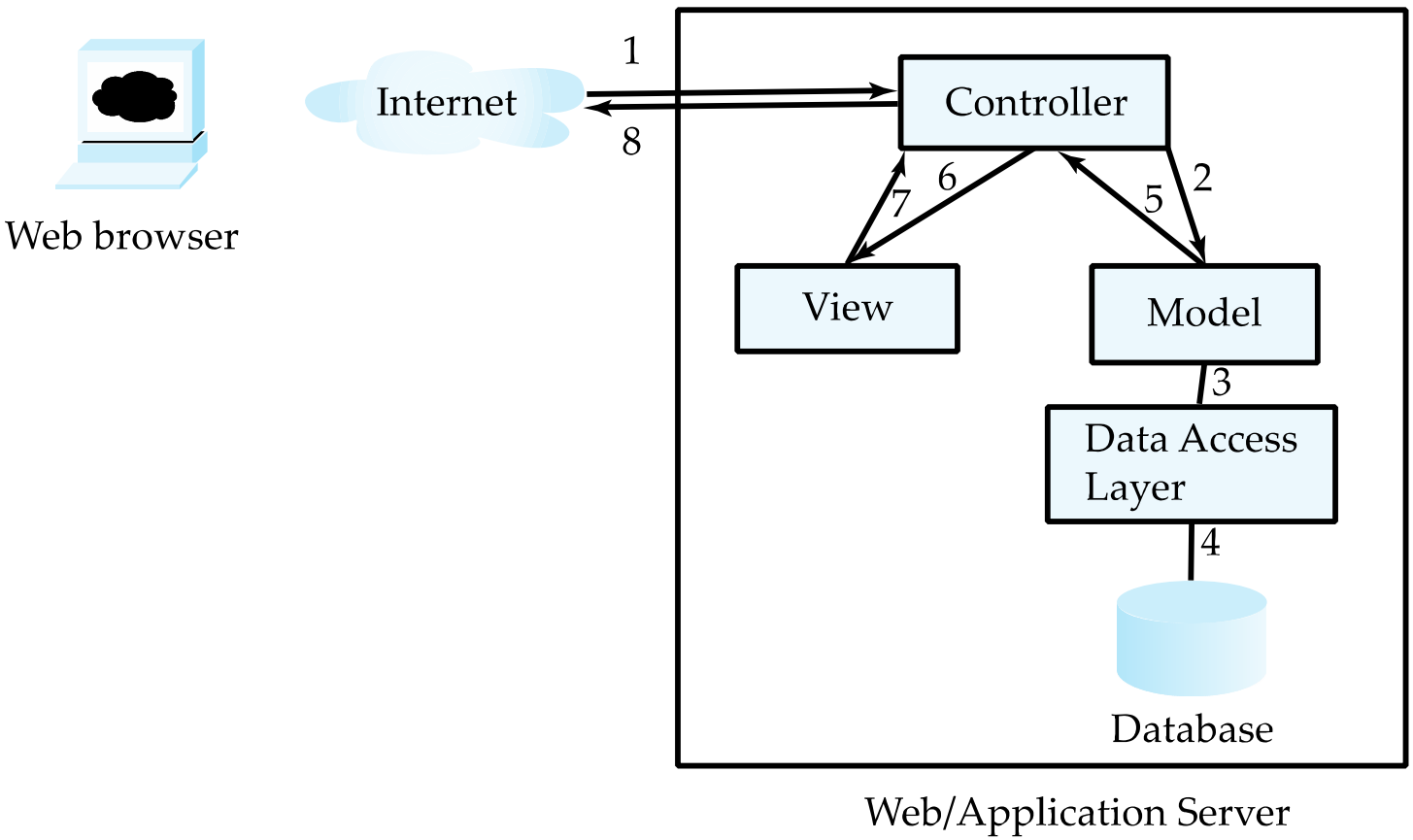


Spatial Queries

- **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
- ..

Notes In Programming With DB

Application Architecture



Application Architectures

- Application layers
  - Presentation or user interface
    - **model-view-controller (MVC)** architecture
      - **model**: business logic
      - **view**: presentation of data, depends on display device
      - **controller**: receives events, executes actions, and returns a view to the user
  - **business-logic** layer
    - provides high level view of data and actions on data
      - often using an object data model
    - hides details of data storage schema
  - **data access** layer
    - interfaces between business logic layer and the underlying database
    - provides mapping from object model of business layer to relational model of database

Web Services

- Allow data on Web to be accessed using remote procedure call mechanism
- Two approaches are widely used
  - **Representation State Transfer (REST)**: allows use of standard HTTP request to a URL to execute a request and return data
    - Returned data is encoded either in XML, or in **JavaScript Object Notation (JSON)**
  - **Big Web Services**:
    - Uses XML representation for sending request data, as well as for returning results
    - Standard protocol layer built on top of HTTP

SQL Injection

- Suppose query is constructed using
  - "select * from instructor where name = ’" + name + "’"
- Suppose the user, instead of entering a name, enters:
  - X’ or ’Y’ = ’Y
- then the resulting statement becomes:
  - "select * from instructor where name = ’" + "X’ or ’Y’ = ’Y" + "’"
  - which is:
    - select * from instructor where name = ’X’ or ’Y’ = ’Y’
  - User could have even used
    - X’; update instructor set salary = salary + 10000; --
- Prepared statement internally uses:
  "select * from instructor where name = ’X\’ or \’Y\’ = \’Y’
- **Always use prepared statements, with user inputs as parameters**

OWASP TOP 10 2021 (The Open Worldwide Application Security Project)

A01:2021	Broken Access Control
A02:2021	Cryptographic Failures
A03:2021	Injection
A04:2021	Insecure Design
A05:2021	Security Misconfiguration
A06:2021	Vulnerable and Outdated Components
A07:2021	Identification and Authentication Failures
A08:2021	Software and Data Integrity Failures
A09:2021	Security Logging and Monitoring Failures
A010:2021	Server-Side Request Forgery

PreparedStatement in JDBC

```
pstmt = conn.prepareStatement( "insert into books
values (?, ?, ?, ?, ?)"); // Five parameters 1 to 5
pstmt.setInt(1, 3001); // Set values for parameters 1
to 5 pstmt.setString(2, "Mahjong 101");
pstmt.setString(3, "Kumar");
pstmt.setDouble(4, 88.88);
pstmt.setInt(5, 88);
int rowAffected = pstmt.executeUpdate();
```


24

PreparedStatement in JDBC

- A PreparedStatement is a pre-compiled SQL statement that is more efficient than calling the same Statement over and over. In a PreparedStatement, '?' indicates a place holder for parameter
- A set of setXxx(*placeHolderNumber*, *value*) methods can be used to fill in the parameters
- **Prevent SQLInjection**

25

JDBC Transaction Example

```
Connection conn = null;
Statement stmt = null;
try {
    conn = DriverManager.getConnection(
        "jdbc:mysql://127.0.0.1:8888/ebookshop", "myuser", "xxxx"); //
    MySQL stmt = conn.createStatement();
    conn.setAutoCommit(false); // Issue two INSERT statements
    stmt.executeUpdate("INSERT INTO books VALUES (5501, 'Peter', 'Mahjong
    101', 5.5, 5)");
    // Duplicate primary key, which triggers a SQLException
    stmt.executeUpdate("INSERT INTO books VALUES (5501, 'Peter', 'More
    Mahjong', 6.6, 6)");
    conn.commit(); // Commit changes only if all statements succeed.
} catch(SQLException ex) {
        System.out.println("-- Rolling back changes --");
        conn.rollback(); // Rollback to the last commit.
        ex.printStackTrace();
} finally { // Step 5: Free resources
        if (stmt != null) stmt.close();
        if (conn != null) conn.close();
}
```

Handling Errors with Exceptions

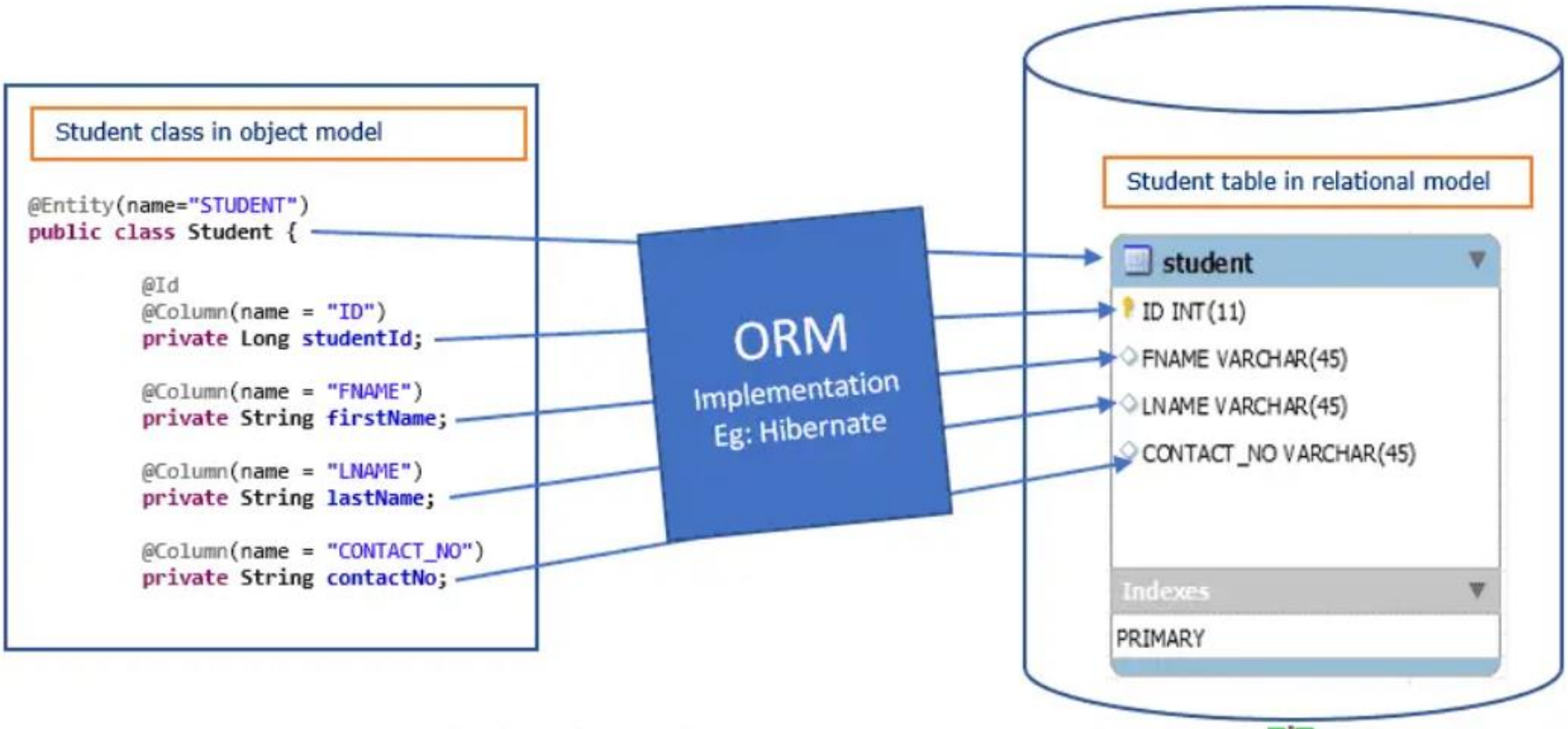
- Programs should recover and leave the database in a consistent state.
- If a statement in the try block throws an exception or warning, it can be caught in one of the corresponding catch statements
- E.g., you could rollback your transaction in a catch { …} block or close database connection and free database related resources in finally {…} block

27

Object-Relational Mapping

- Allows application code to be written on top of object-oriented data model, while storing data in a traditional relational database
- Schema designer has to provide a mapping between object data and relational schema
  - E.g., Java class *Student* mapped to relation *student,* with corresponding mapping of attributes
  - An object can map to multiple tuples in multiple relations
- Query can be run to retrieve objects satisfying specified predicates

Object-Relational Mapping



ORM implements responsibility of mapping the Object to Relational Model.

JavaByDeveloper

Object-Relational Mapping and Hibernate (Cont.)

- The **Hibernate** object-relational mapping system is widely used
  - Public domain system, runs on a variety of database systems
  - Supports a query language that can express complex queries involving joins
    - Translates queries into SQL queries
  - Allows relationships to be mapped to sets associated with objects
    - E.g., courses taken by a student can be a set in Student object

- https://www.geeksforgeeks.org/hibernate-example-using-jpa-and-mysql/

Object-Relational Mapping and Hibernate (Cont.)

- The **Entity Data Model** developed by Microsoft
  - Provides an entity-relationship model directly to application
  - Maps data between entity data model and underlying storage, which can be relational
  - Entity SQL language operates directly on Entity Data Model

THANKS YOU