Introduction to Relational Database Management Systems

Maria K. Krommyda
mariakr@dblab.ece.ntua.gr

Outline

- RDBMS History
- Relational Model Overview
- RDBMS Overview
- Integrity Constraints in RDBMS
- Views
- Triggers
- Client/Server Database Model
- JDBC
- Microsoft SQL Server

RDBMS History – The Ancestors

Early 1960's

- IDS (Integrated Data Store)
  - The first DBMS
  - Network data model (Directed acyclic graph with nodes & edges)
  - Charles Bachman @ Honeywell Information Systems
  - 1973 ACM Turing Award "For his outstanding contributions to database technology"

Mid 1960's

- IMS (Information Management System)
  - The first commercially DBMS
  - IBM
  - Hierarchical model (Tree-based Representation)

RDBMS History – The Relational Model

1970

- Relational Model
  - Edgar (Ted) Codd @ IBM San Jose Lab
  - “A Relational Model of Data for Large Shared Data Banks”
  - 1981 ACM Turing Award “For his fundamental and continuing contributions to the theory and practice of database management systems, esp. relational databases”

RDBMS History – The First RDBMSs

Late 1970’ s

- INGRES
  - University of California, Berkeley
  - Michael Stonebraker & Eugene Wong
  - Used QUEL as its query language
  - Similar to System R, but based on different hardware and operating system
  - Became commercial and followed up POSTGRES which was incorporated into Informix.

- System R
  - IBM San Jose Lab
  - Structured Query Language (SQL)
  - Evolved into SQL/DS which later became DB2

(R)DBMS History – Important Dates

- **1976:** Peter Chen defined the Entity-Relationship (ER) model
- **1985:** Object-oriented DBMS (OODBMS).
- **90s:** Incorporation of object-orientation in RDBMS
- **1991:** Microsoft Access, a personal DBMS
- **Mid 90s:** First Internet database applications
- **Late 90s:** XML used in DBMS
- **Early 00s:** RDF used in DBMS

RDBMS History – Today

- The main players
  - **Oracle**
    - Oracle Database & MySQL (earlier MySQL AB, Sun)
  - **IBM**
    - DB2
  - **Microsoft**
    - SQL Server

Relational Model – Basic Concepts

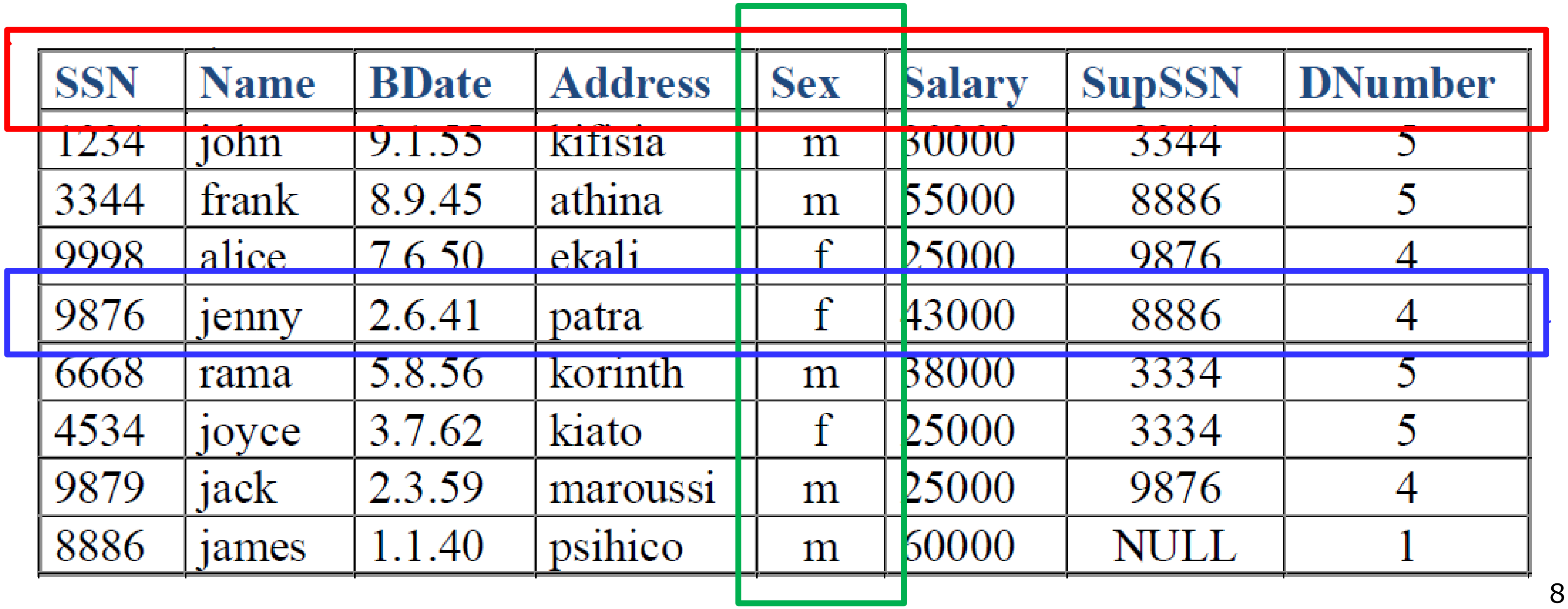
- Data is represented as mathematical n-ary relations
- Table is a relation representation

- Relation (table) basic concepts:

Attribute
(column)

Attributes
Names

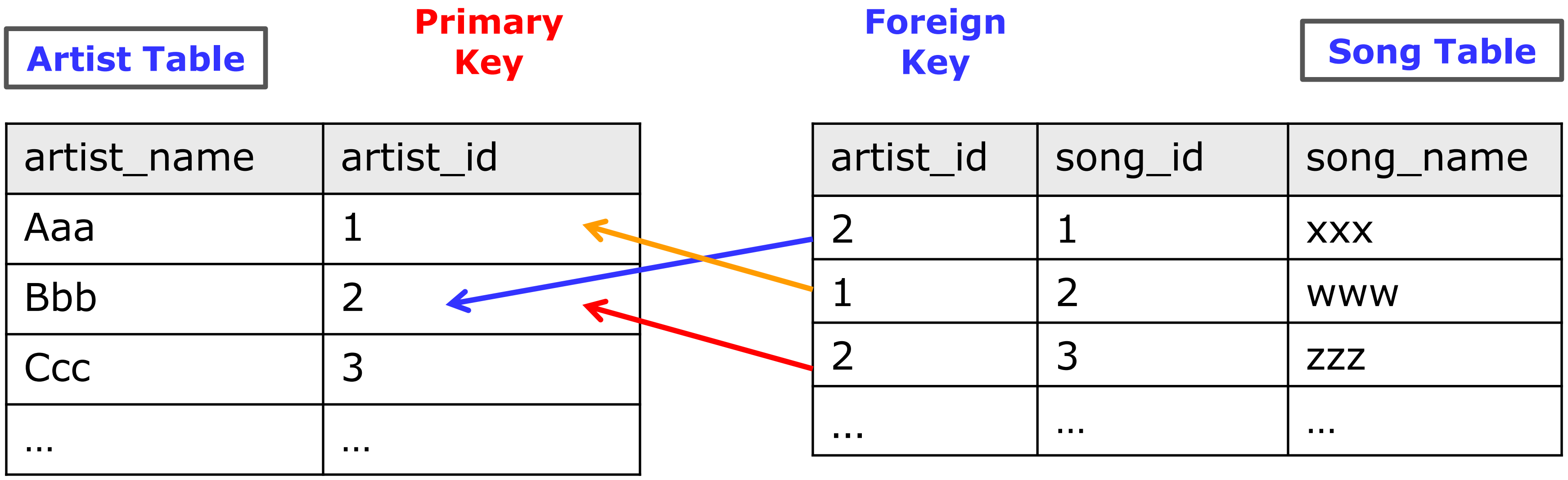
Tupple
(row)



SSN	Name	BDate	Address	Sex	Salary	SupSSN	DNumber
1234	john	9.1.55	kifisia	m	30000	3344	5
3344	frank	8.9.45	athina	m	55000	8886	5
9998	alice	7.6.50	ekali	f	25000	9876	4
9876	jenny	2.6.41	patra	f	43000	8886	4
6668	rama	5.8.56	korinth	m	38000	3334	5
4534	joyce	3.7.62	kiato	f	25000	3334	5
9879	jack	2.3.59	maroussi	m	25000	9876	4
8886	james	1.1.40	psihico	m	60000	NULL	1

Relational Model – Relations (1/2)

- Relating Relations…



Artist Table

Primary Key

Foreign Key

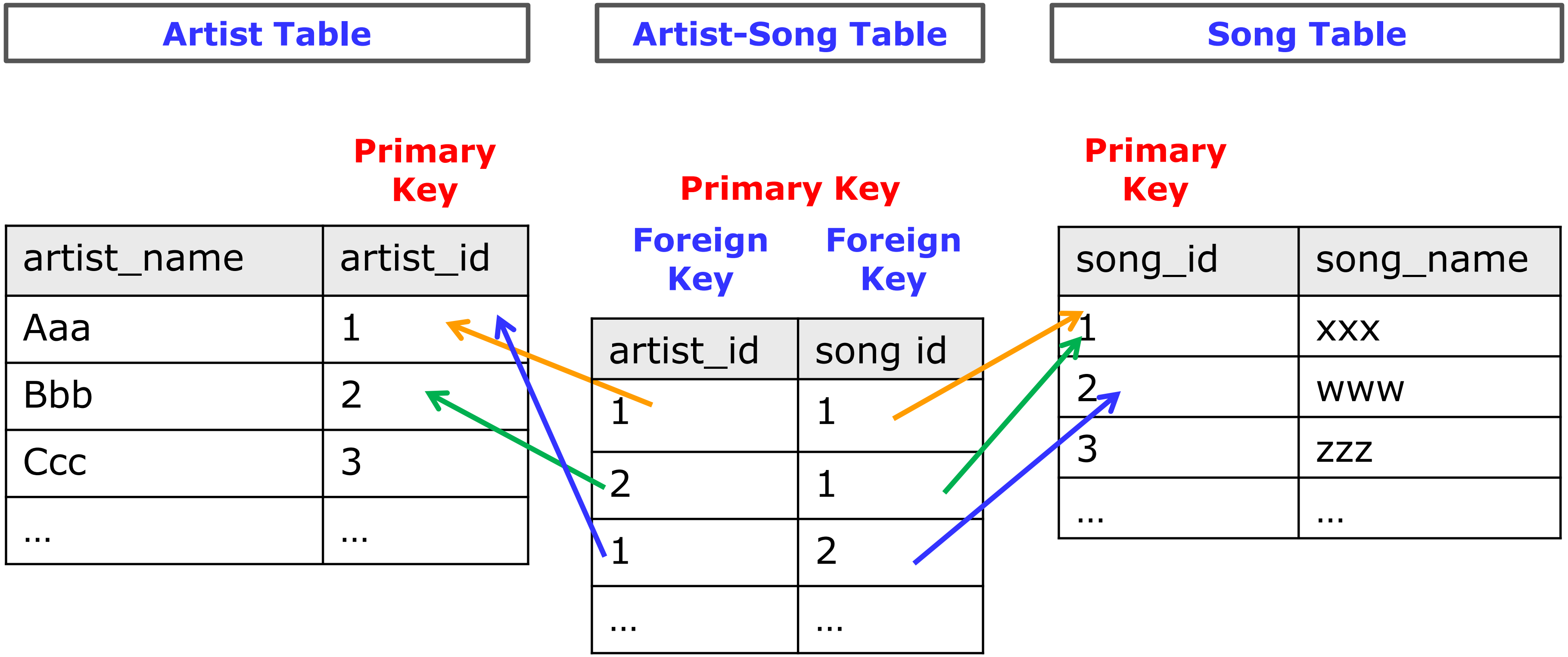
Song Table

artist_name	artist_id
Aaa	1
Bbb	2
Ccc	3
…	…

artist_id	song_id	song_name
2	1	xxx
1	2	www
2	3	zzz
…	…	…

- Limitations?

Relational Model – Relations (2/2)



Artist Table

Primary Key

artist_name	artist_id
Aaa	1
Bbb	2
Ccc	3
…	…

Artist-Song Table

Primary Key

Foreign Key **Foreign Key**

artist_id	song id
1	1
2	1
1	2
…	…

Song Table

Primary Key

song_id	song_name
1	xxx
2	www
3	zzz
…	…

RDBMS Overview – Basic Objects

- Tables
- Views
- Triggers
- Stored Procedures
- Functions
- Rules
- Cursors

RDBMS Overview – Data Types

- bit: boolean number
- int, smallInt, bigInt, tinyInt: Integer number
- decimal, numeric: Real numbers
- char, varchar, nchar, nvarchar, text: Strings
- date, datetime: Date and time
- money, smallmoney: money values
- binary: Images and other large objects
- …

RDBMS Overview – Operators

- Arithmetic: +, -, *, /, %
- Assignment: =
- Comparison: <, >, <=, >= <>, =, !=, !<, !>
- Logical: AND, OR, NOT, IN, LIKE, BETWEEN, ANY, ALL, EXISTS, SOME
- String: Concatenation (+)
- Unary: -, +, ~
- Bitwise: &, |, ^
- …

RDBMS Overview – Operations (1/3)

- Database Level
  - Defining “working” database

    Use <dbname>
  - Creating a database

    Create database <dbname>
  - Deleting a database

    Drop database <dbname>

RDBMS Overview – Operations (2/3)

- Schema Level
  - Create Table
  - Drop Table
  - Alter Table (Used to modify table structure)
    - Add new column
    - Change data type of existing column
    - Delete a column
    - Add or remove constraints like foreign key, primary key

RDBMS Overview – Operations (2/3)

Create Table Example

```
CREATE TABLE Person(
        personID integer,
        FirstName varchar(15) not null,
        LastName varchar(20),
        Age demical(3,1),
        orgID integer,
        primary key (personID)
        foreign key orgID references Organization.ID
);
```

RDBMS Overview – Operations (2/3)

Drop/Alter Table Examples

- **DROP TABLE** Person;

- **ALTER TABLE** Person **ADD** Email **varchar**(30);
- **ALTER TABLE** Person **ADD** (Email **varchar**(30), Telephone **varchar**(20));

- **ALTER Table** Person **DROP COLUMN** Age;

- **ALTER TABLE** Person **ALTER COLUMN** LastName **varchar**(50);

- **ALTER TABLE** Person **ADD CONSTRAINT** const_LastName **UNIQUE** (LastName);

- **ALTER TABLE** Person **ADD** Email **varchar**(30) **NOT NULL**;

RDBMS Overview – Operations (3/3)

- Data Level
  - Select
  - Insert
  - Update
    - Update data to all/selected columns/rows
  - Delete
    - Delete all/selected rows from table

Integrity Constraints in RDBMS

- Integrity constraints are used to ensure accuracy and consistency of data in a relational database.

- Types
  - Entity integrity => Primary Key
  - Referential Integrity => Foreign Key
  - Domain Integrity

  - User Defined Integrity

Integrity Constraints in RDBMS – Entity Integrity

- Every table must have a primary key
- Primary key should be unique and not null
- Used: Insertions and Updates
- SQL
  - PRIMARY KEY
  - UNIQUE (Candidate Keys)
- Primary keys
  - Referenced by Foreign keys
  - Indexes

Creating Unique Values in RDBMS

- MS SQL Server
  - Identity (seed, increment)
  - Seed is the initial value
  - Increment is the value by which we need to skip to fetch the next value
  - Identity(1,2) will generate sequence numbers 1,3,5,7…

- MySQL
  - AUTO_INCREMENT
  - The starting value is 1, and it will increment by 1 for each new record.
  - AUTO_INCREMENT = k (start from k value)

Integrity Constraints in RDBMS – Referential Integrity

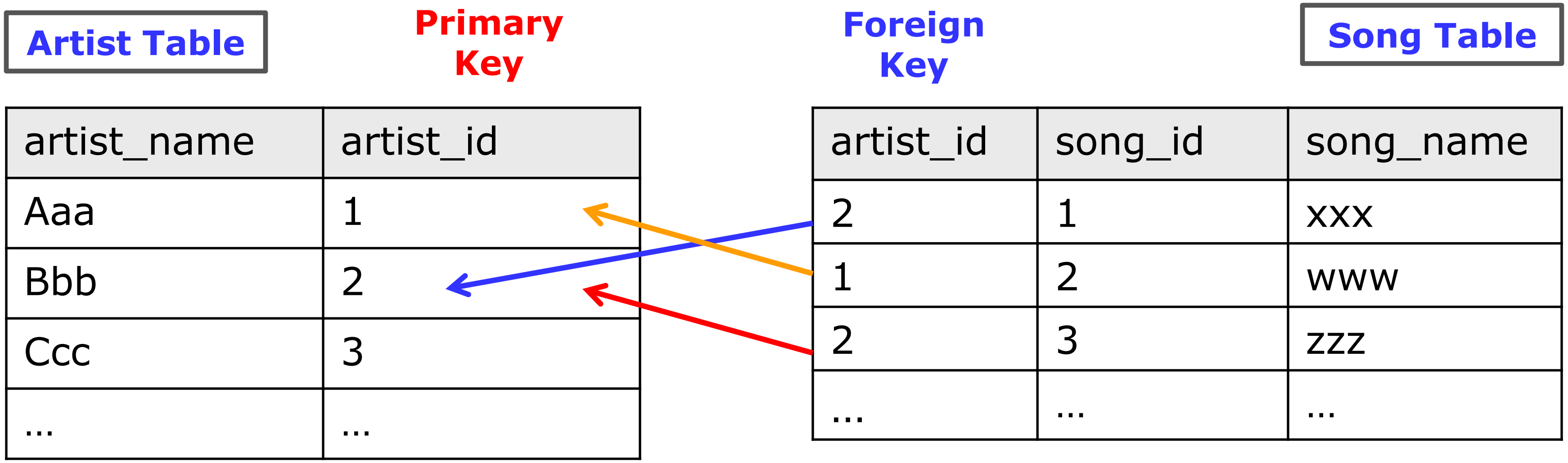
- The referential integrity constraint, states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.

  Foreign Key value ⇒ Primary Key value

- Referential Integrity in SQL

  pk type **PRIMARY KEY**

  FOREIGN KEY *fk* **REFERENCES** *pk*



Artist Table

Primary Key

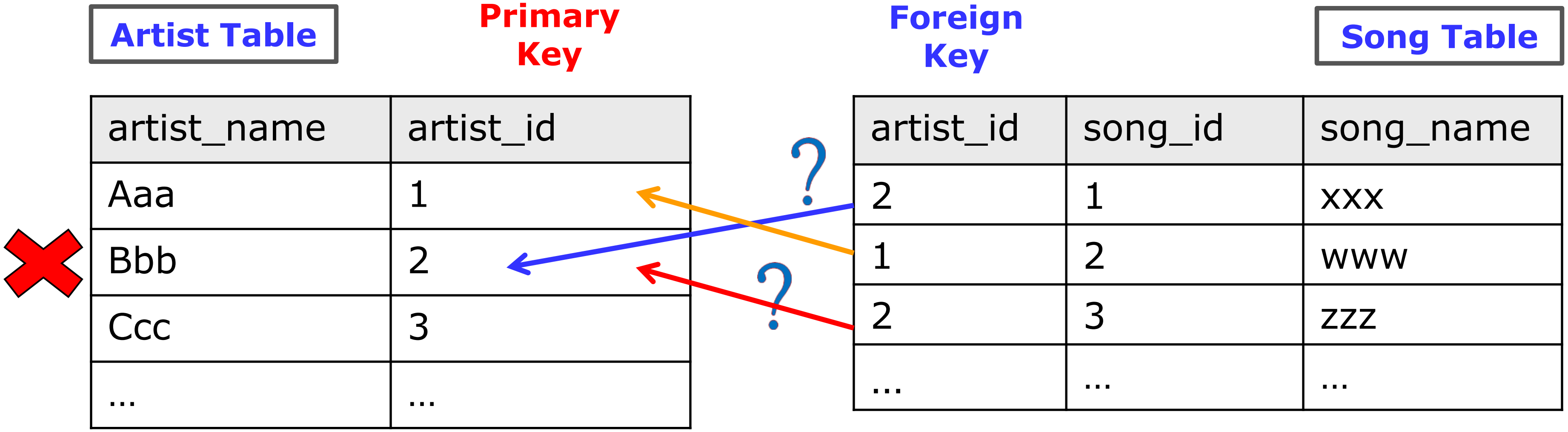
artist_name	artist_id
Aaa	1
Bbb	2
Ccc	3
…	…

Foreign Key

Song Table

artist_id	song_id	song_name
2	1	xxx
1	2	www
2	3	zzz
…	…	…

Integrity Constraints in RDBMS – Referential Integrity Example



Artist Table

Primary Key

artist_name	artist_id
Aaa	1
Bbb	2
Ccc	3
…	…

Foreign Key

Song Table

artist_id	song_id	song_name
2	1	xxx
1	2	www
2	3	zzz
…	…	…

- Delete tuple (2, Bbb) ✗
- Possible scenarios ? ? ?
  - Reject
  - Set Song.artist_id = null
  - Delete Song tuples

Integrity Constraints in RDBMS – Referential Integrity Constraints in SQL

CREATE TABLE *a* (

.....

FOREIGN KEY *fk* **REFERENCES** *pk* **action**

.....)

Where **action** is:

- nothing or NO ACTION (deletion/update rejected)
- ON DELETE SET NULL / ON UPDATE SET NULL
- ON DELETE CASCADE / ON UPDATE CASCADE

Integrity Constraints in RDBMS – Domain Integrity

- Column (attribute) Constraints
  - NOT NULL
  - CHECK (e.g., CHECK(age >= 0))
- Domain Constraints
  - Use Column Constraints
  - Similar to user-defined datatypes
  - Reusability
  - "Programmer friendly" (gives names)
- Used: Insertions and Updates

Integrity Constraints in RDBMS – Domain Integrity Example

- Define Domain Constraint

```
CREATE DOMAIN validAge INT (
            CONSTRAINT positive CHECK (VALUE >= 0),
            CONSTRAINT limit CHECK (VALUE < 150 ),
            CONSTRAINT not-null-value CHECK( VALUE NOT NULL));
```

- Use Domain Constraint

```
CREATE TABLE Employee (
        ....
        age validAge,
)
```

Views Intro

- View is a virtual table
- Create View SQL syntax

```
CREATE VIEW view_name [(view_columns)]
AS          SQL Query
```

- View contents are specified by the View definition
- View contains rows and columns, just like a real table
- A View can defined over several tables or other views
- A View may define different/new attributes
- If a change occurs in the tables it is reflected into the view
- Queries over Views are the same as queries over relations
- Updates under several restrictions

Updatable Views

- Updatable:
  - The **from** clause has only one database relation.
  - The **select** clause contains only attribute names of the relation, and does not have any expressions, aggregates, or distinct specification.
  - Any attribute **not listed** in the select clause **can be set to *null***; *that is, it does* not have a not null constraint and is not part of a primary key.
  - The query does not have a group by or having clause.
  - The **where** clause may have restrictions. ?

Views Examples

```
CREATE VIEW OLD_PERSONS AS
        select *
        from Person
        where Age > 80;
```

```
CREATE VIEW OLD_PERSONS_NAMES (onoma) AS
        select FirstName
        from Person
        where Age > 80;
```

Views vs. Tables

- Views can represent a subset (or "superset") of the data contained in a table
- Views can join or simplify multiple tables
- Views can act as aggregated tables (sum, average etc.) and present the calculated results
- Views require very little storing space (only the definition of the view)
- Views can limit the degree of exposure of data to the outer world (Users groups)
- Views allow application interoperability through columns renaming/rearranging

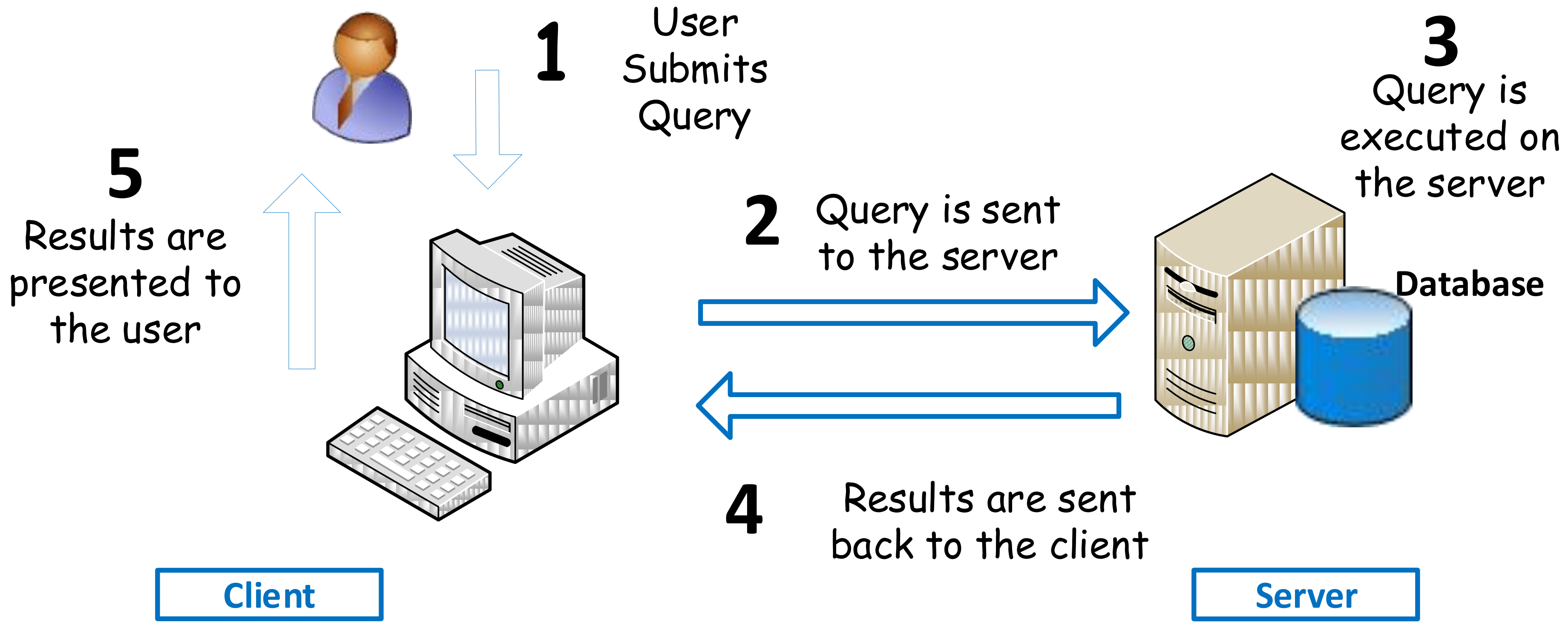
Triggers Intro

- A Trigger is procedural code that is automatically executed in response to certain events on a particular table or view
- Triggers are stored in, and managed by the RDBMS
- Each trigger is attached to a single specified table/view
- Triggers Events: insert, update, delete
- Using triggers, data integrity problems can be eliminated
- Triggers can access and/or modify other tables
- Triggers can executed
  - Before a specified event
  - After a specified eventк

Triggers Example

```
CREATE TRIGGER Books_Delete
AFTER DELETE ON Books
    REFERENCING OLD ROW AS Old
FOR EACH ROW
    INSERT INTO Books_Deleted_Log
        VALUES (Old.title);
```

Client / Server Database Model



1 User Submits Query

2 Query is sent to the server

3 Query is executed on the server

4 Results are sent back to the client

5 Results are presented to the user

Database

Client

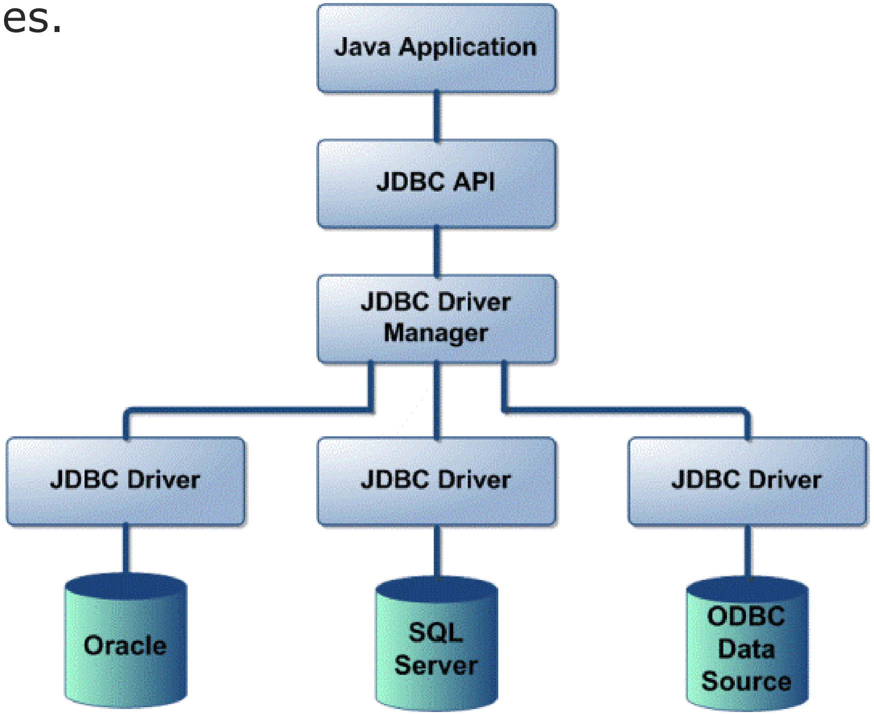
Server

JDBC Intro

- JDBC (Java Database Connectivity)
- An API for the Java programming language that defines how a client interact with a database.
- JDBC works with Java on a variety of platforms, e.g., Windows, Mac OS, and the various versions of UNIX.

JDBC Architecture

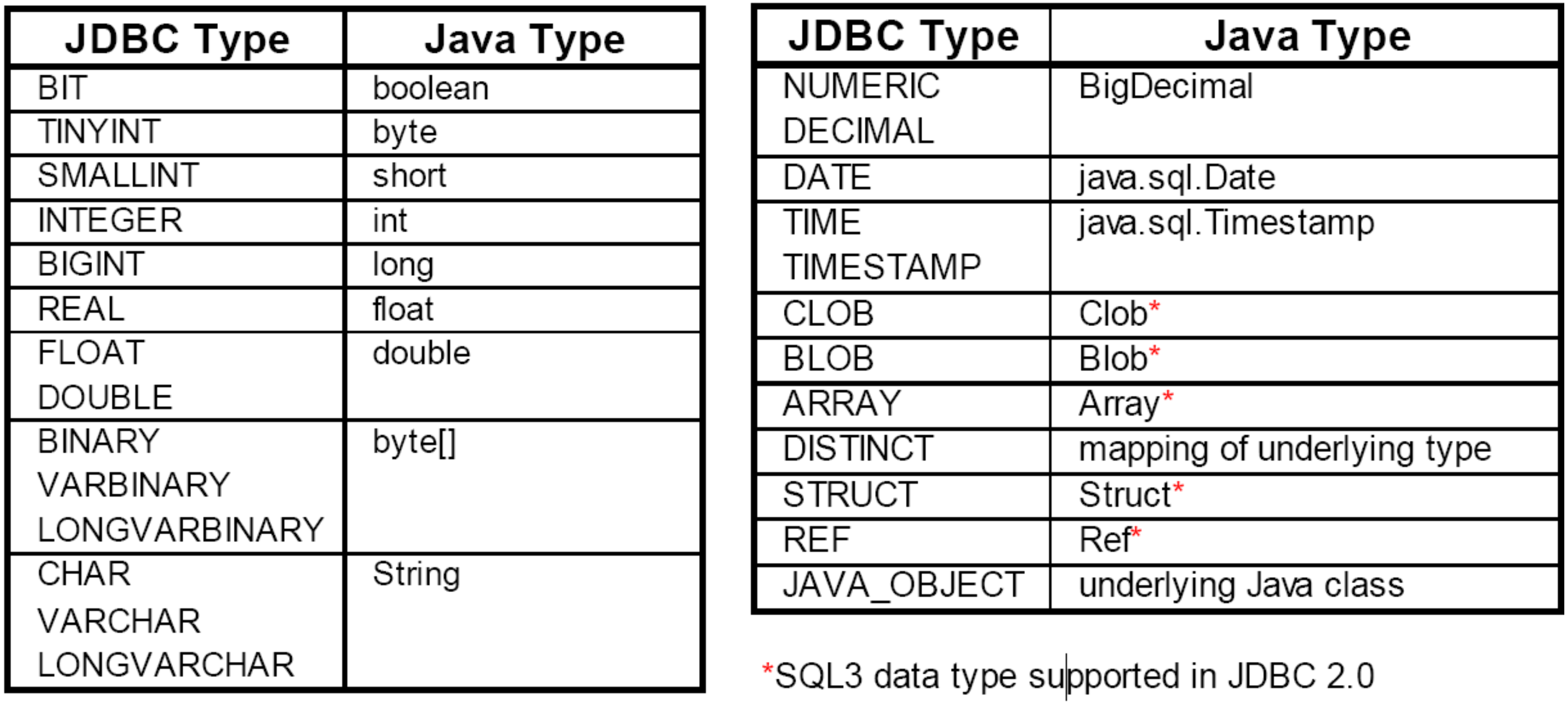
- Two layers Architecture
  - **JDBC API:** Java Application to JDBC Driver Manager
  - **JDBC Driver API:** JDBC Driver Manager to (database-specific) Driver
    - Ensures that the correct driver is used to access each data source.
    - Multiple concurrent drivers connected to multiple heterogeneous databases.



JDBC Basic Steps

- Seven steps in querying databases
  1. Load the JDBC driver
  2. Define the connection URL
  3. Establish the connection
  4. Create a statement object
  5. Execute a query or update
  6. Process the results
  7. Close the connection

JDBC vs. Java Data types



JDBC Type	Java Type
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT DOUBLE	double
BINARY VARBINARY LONGVARBINARY	byte[]
CHAR VARCHAR LONGVARCHAR	String

JDBC Type	Java Type
NUMERIC DECIMAL	BigDecimal
DATE	java.sql.Date
TIME TIMESTAMP	java.sql.Timestamp
CLOB	Clob*
BLOB	Blob*
ARRAY	Array*
DISTINCT	mapping of underlying type
STRUCT	Struct*
REF	Ref*
JAVA_OBJECT	underlying Java class

*SQL3 data type supported in JDBC 2.0

Basic JDBC Components

- **Connection:** connection objects are used to communication with database.

- **Statement:** Statement objects used to submit the SQL statements to the database.

- **ResultSet:** These objects hold data retrieved from a database after you execute an SQL query using Statement objects.

- **ResultSetMetaData:** Info regarding Result set object (e.g., number of columns, columns types, etc.)

Statement Methods

- boolean **execute**(String SQL)
  - Execute SQL statements.
  - Returns true if a ResultSet object can be retrieved; otherwise, it returns false.
- ResultSet **executeQuery**(String SQL)
  - Use this method when you expect to get a result set, as you would with a SELECT statement.
  - Returns a ResultSet object.
- int **executeUpdate**(String SQL)
  - Used for executing INSERT, UPDATE, or DELETE SQL statements
  - Returns the numbers of rows affected by the execution of the SQL statement.

ResultSet Methods

- boolean **first**()
  - Moves the cursor to the first row
- void **last**()
  - Moves the cursor to the last row.
- boolean **previous**()
  - Moves the cursor to the previous row
- boolean **next**()
  - Moves the cursor to the next row
- int **getRow**()
  - Returns the row number that the cursor is pointing to.
- int **getXXX**(String columnName)
  - Returns the value in the current row in the column named columnName
  - Where **XXX** is int, float, long, String, etc.
- int **getXXX**(int columnIndex)
  - Returns the value in the current row in the specified column index.
  - The column index starts at 1
  - Where **XXX** is int, float, long, String, etc.

Database Example

```
CREATE DATABASE dbTest

CREATE TABLE Employee (
        ID int PRIMARY KEY,
        Name varchar(40),
        Salary demical(10,2)
)
```

Use ConnectSQLServer.java to access dbTest Database

ConnectSQLServer.java

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class ConnectSQLServer {
    public static void main(String[] args) {
        try {
                Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
                 Connection connection = DriverManager.getConnection(
                           "jdbc:sqlserver://localhost:1433;databaseName=dbTest","myUserName", "myPassword");

                Statement statement = connection.createStatement();
                String queryString = “Select Name, Salary from Employee";
                ResultSet resultSet = statement.executeQuery(queryString);

                while (resultSet.next()) {
                       System.out.println("Employee Name:" + rs.getString("Name") );
                       System.out.println("Employee Salary:" + rs.getFloat("Salary") ) ;
                                                                  //rs.getBigDecimal(“Salary”,2);
                }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Microsoft SQL Server

- MS SQL Server
  - Database server
  - Product of Microsoft
  - Relational DB
- From: 1989 (SQL Server 1.0)

  To: July 2011 (SQL Server 2008 R2)
- Runs on: Windows 7, Vista, Server (03&08), XP, ME, 98
- Platform: 32 & 64

- SQL Server & MySQL Installation Guides

  http://www.cslab.ntua.gr/courses/db/links.go

Project Implementation

- Linux/7/Vista/Win2000/XP/2003/98/ME …
- SQL Server 2000/2005/2008/postgres/mysql …
- JAVA, VB.NET, PYTHON, C++ ….

Project Requirements

- Database Design
- Database Design
- Database Design
  - **Use Integrity Constraints !!!**

- Define meaningful Queries, Views, etc.

- Graphical User Interface
  - **Fully functional**
    - **View DB**
    - **Insert DB**
    - **Query DB**
    - **etc.**
  - **User-friendly**
    - **Drop-down list**
    - **Radio button**
    - **Check box**
    - **etc.**

Thank you