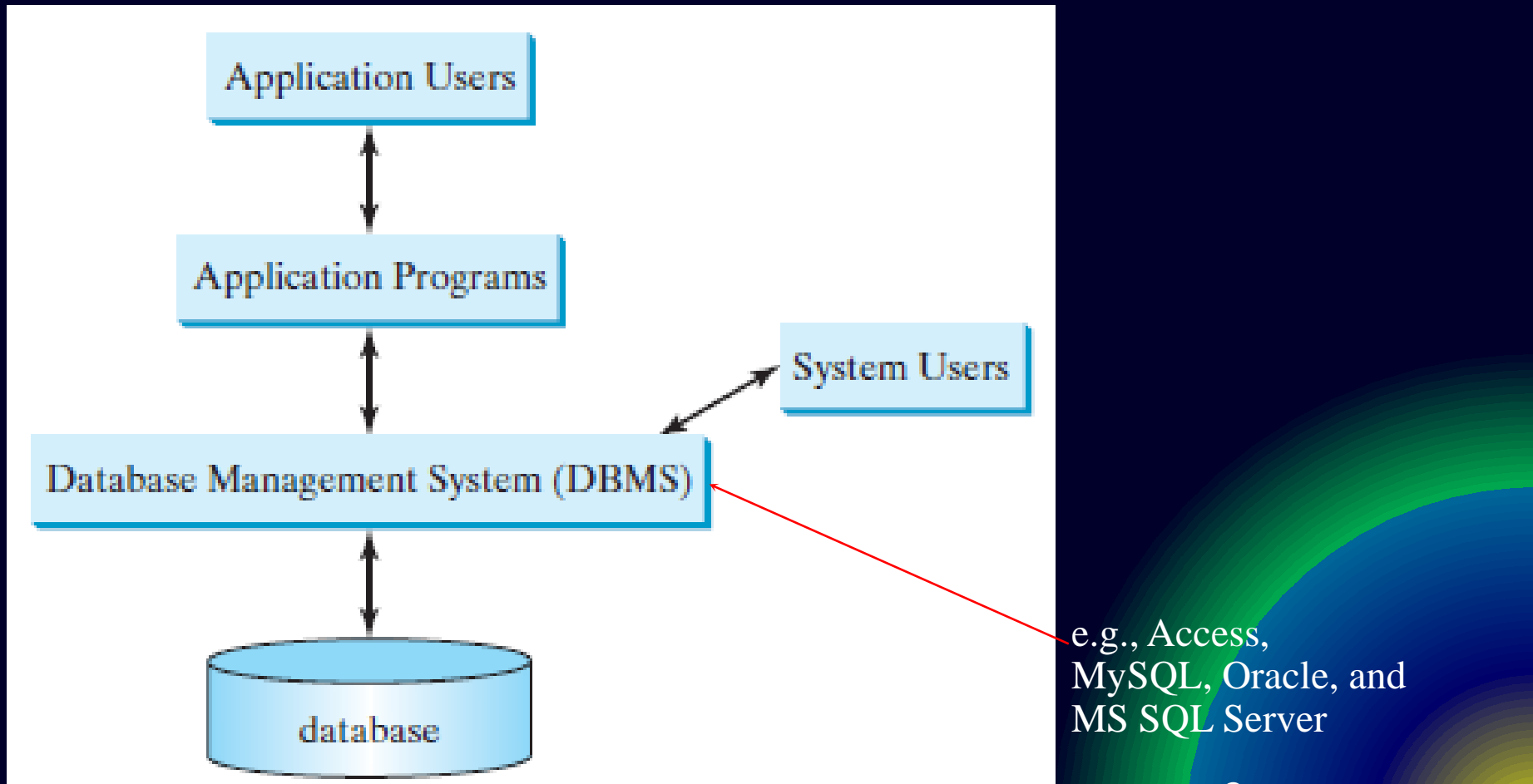


Java Database Programming (Chapter 32)

What is a Database System?

A **database system** consists of a database, the software that stores and manages data in the database, and the application programs that present data and enable the user to interact with the database system,



Relational Data Model

- Most of today's database systems are **relational database systems**, based on the relational data model.
- A relational data model has three key components: **structure**, **integrity** and **languages**.
 - *Structure* defines the representation of the data.
 - *Integrity* imposes constraints on the data.
 - *Language* provides the means for accessing and manipulating data.

Relational Structure

- A **relational database** consists of a set of relations.
- A relation has two things in one: a *schema* and an *instance* of the schema.
- The schema defines the relation and an instance is the content of the relation at a given time.
- An instance of a relation is nothing more than a table with rows and named columns.
- For convenience, we refer to instances of relations as just *relations or tables*.

Relational Structure

A row of a table represents a **record**, and a column of a table represents the value of a **single attribute of the record**. In relational database theory, a row is called a **tuple** and a column is called an **attribute**.

Relation/Table Name	Columns/Attributes				
Course Table	courseId	subjectId	courseNumber	title	numOfCredits
Tuples/ Rows	11111	CSCI	1301	Introduction to Java I	4
	11112	CSCI	1302	Introduction to Java II	3
	11113	CSCI	3720	Database Systems	3
	11114	CSCI	4750	Rapid Java Application	3
	11115	MATH	2750	Calculus I	5
	11116	MATH	3750	Calculus II	5
	11117	EDUC	1111	Reading	3
	11118	ITEC	1344	Database Administration	3

Relational Structure

- Tables describe the relationship among data.
- Each row in a table represents a record of related data. For example, “11111”, “CSCI”, “1301”, “Introduction to Java I”, and “4” are related to form the first record in the Course table.
- Just as data in the same row are related, so too data in different tables may be related through common attributes.
- For example, the Course table and the Enrollment table in the next slide are related through their common attribute courseId, and the Enrollment table and the Student table are related through ssn.

Course and Enrollment Tables

Enrollment Table

ssn	courseId	dateRegistered	grade
444111110	11111	2004-03-19	A
444111110	11112	2004-03-19	B
444111110	11113	2004-03-19	C
...			

Course Table

courseId	subjectId	courseNumber	title	numOfCredits
11111	CSCI	1301	Introduction to Java I	4
11112	CSCI	1302	Introduction to Java II	3
11113	CSCI	3720	Database Systems	3
...				

Student and Enrollment Tables

Student Table									
ssn	firstName	mi	lastName	phone	birthDate		street	zipCode	deptID
444111110	Jacob	R	Smith	9129219434	1985-04-09	99	Kingston Street	31435	BIOL
444111111	John	K	Stevenson	9129219434	null	100	Main Street	31411	BIOL
444111112	George	K	Smith	9129213454	1974-10-10	1200	Abercorn St.	31419	CS
444111113	Frank	E	Jones	9125919434	1970-09-09	100	Main Street	31411	BIOL
444111114	Jean	K	Smith	9129219434	1970-02-09	100	Main Street	31411	CHEM
444111115	Josh	R	Woo	7075989434	1970-02-09	555	Franklin St.	31411	CHEM
444111116	Josh	R	Smith	9129219434	1973-02-09	100	Main Street	31411	BIOL
444111117	Joy	P	Kennedy	9129229434	1974-03-19	103	Bay Street	31412	CS
444111118	Toni	R	Peterson	9129229434	1964-04-29	103	Bay Street	31412	MATH
444111119	Patrick	R	Stoneman	9129229434	1969-04-29	101	Washington St.	31435	MATH
444111120	Rick	R	Carter	9125919434	1986-04-09	19	West Ford St.	31411	BIOL

Enrollment Table			
ssn	courseId	dateRegistered	grade
444111110	11111	2004-03-19	A
444111110	11112	2004-03-19	B
444111110	11113	2004-03-19	C
444111111	11111	2004-03-19	D
444111111	11112	2004-03-19	F
444111111	11113	2004-03-19	A
444111112	11114	2004-03-19	B
444111112	11115	2004-03-19	C
444111112	11116	2004-03-19	D
444111113	11111	2004-03-19	A
444111113	11113	2004-03-19	A
444111114	11115	2004-03-19	B
444111115	11115	2004-03-19	F
444111115	11116	2004-03-19	F
444111116	11111	2004-03-19	D
444111117	11111	2004-03-19	D
444111118	11111	2004-03-19	A
444111118	11112	2004-03-19	D
444111118	11113	2004-03-19	B

Table vs. File

NOTE:

A table or a relation is not same as a file.
Most of the relational database systems
store multiple tables in a file.

Integrity Constraints

- An **integrity constraint** imposes a condition that all legal instances of the relations must satisfy.
- In general, there are three types of constraints: *domain constraint*, *primary key constraint*, and *foreign key constraint*.
- Domain constraints and primary key constraints are known as *intra-relational constraints*, meaning that a constraint involves only one relation.
- The foreign key constraint is known as *inter-relational*, meaning that a constraint involves more than one relation.

Domain Constraints

- *Domain constraints* specify the permissible values for an attribute.
- Domains can be specified using standard data types such as integers, floating-point numbers, fixed-length strings, and variant-length strings.
- Additional constraints can be specified to narrow the ranges.
- You can also specify whether an attribute can be null.

Primary Key Constraints

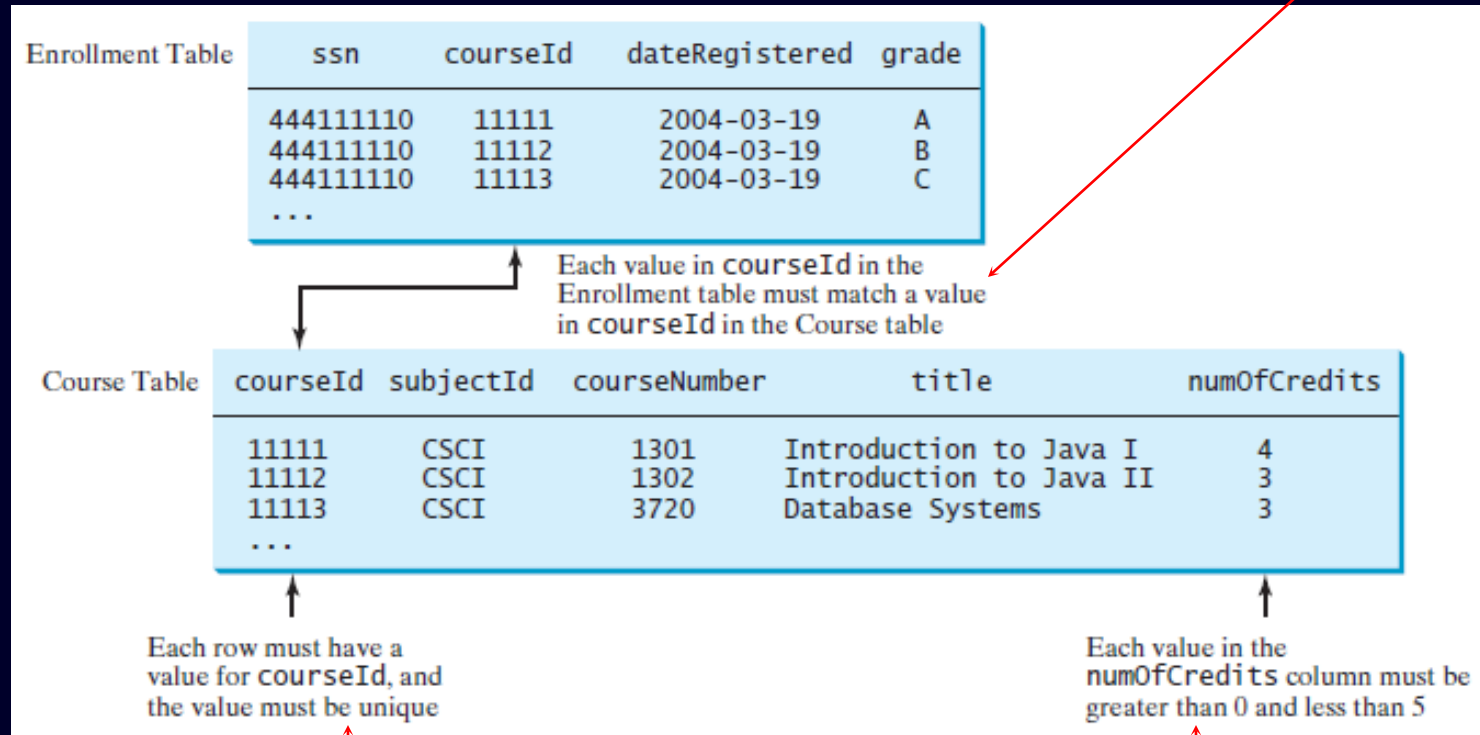
- The *primary key constraint* specifies that the primary key value of a tuple cannot be null and no two tuples in the relation can have the same value on the primary key.
- The DBMS enforces the primary key constraint.
- For example, if you attempt to insert a record with the same primary key as an existing record in the table, the DBMS would report an error and reject the operation.

Foreign Key Constraints

- In a relational database, data are related.
- Tuples in a relation are related and tuples in different relations are related through their common attributes.
- Informally speaking, the common attributes are foreign keys.
- The *foreign key constraints* define the relationships among relations.

Integrity Constraints

*Foreign key
constraint*



*Primary key
constraint*


*domain
constraint*

Foreign Key Constraints Formal Definition

Formally, a set of attributes FK is a *foreign key* in a relation R that references relation T if it satisfies the following two rules:

- The attributes in FK have the **same domain** as the primary key in T .
- A **non-null** value on FK in R must match a primary key value in T .

SQL

- **Structured Query Language (SQL)** is the language for defining tables and integrity constraints and for accessing and manipulating data.
 - It is the universal language for accessing relational database systems.
- 

Examples of simple SQL statements

Create table

Drop table

Select

Insert

Delete

Update

```
create table Course (  
    courseId char(5),  
    subjectId char(4) not null,  
    courseNumber integer,  
    title varchar(50) not null,  
    numOfCredits integer,  
    primary key (courseId)  
);
```

```
create table Student (  
    ssn char(9),  
    firstName varchar(25),  
    mi char(1),  
    lastName varchar(25),  
    birthDate date,  
    street varchar(25),  
    phone char(11),  
    zipCode char(5),  
    deptId char(4),  
    primary key (ssn)  
);
```

Examples of simple SQL statements

Creating a Table with Foreign Keys

```
create table Enrollment (  
    ssn char(9),  
    courseId char(5),  
    dateRegistered date,  
    grade char(1),  
    primary key (ssn, courseId),  
    foreign key (ssn) references Student,  
    foreign key (courseId) references  
Course)  
;
```

Examples of simple SQL statements

Create table

Drop table

Select

Insert

Delete

Update

```
drop table Enrollment;
```

```
drop table Course;
```

```
drop table Student;
```

Examples of simple SQL statements

Create table

Drop table

Select

Insert

Delete

Update

```
select firstName, mi, lastName  
from Student  
where deptId = 'CS';
```

```
select firstName, mi, lastName  
from Student  
where deptId = 'CS' and zipCode = '31411';
```

```
select *  
from Student  
where deptId = 'CS' and zipCode = '31411';
```

Examples of simple SQL statements

Create table

Drop table

Select

Insert

Delete

Update

```
insert into Course (courseId, subjectId, courseNumber, title,  
                   numOfCredits)  
values ('11113', 'CSCI', '3720', 'Database Systems', 3);
```

Examples of simple SQL statements

Create table

Drop table

Select

Insert

Update

Delete

```
update Course
set numOfCredits = 4
where title = 'Database Systems';
```

Examples of simple SQL statements

Create table

Drop table

Select

Insert

Update

Delete

```
delete from Course  
where title = 'Database System';
```

Displaying Sorted Tuples


SQL provides the **order by** clause to sort the output using the following syntax:

```
select column-list  
from table-list  
[where condition]  
[order by columns-to-be-sorted];
```

- **columns-to-be-sorted** specifies a column or a list of columns to be sorted. By default, the order is **ascending**. To sort in a descending order, append the **desc** keyword.
- You could also append the **asc** keyword after **columns-to-be-sorted**, but it is not necessary.
- When multiple columns are specified, the rows are sorted based on the first column, then the rows with the same values on the first column are sorted based on the second column, and so on.

Displaying Sorted Tuples

```
select lastName, firstName, deptId  
from Student  
where deptId = 'CS'  
order by lastName desc, firstName asc;
```

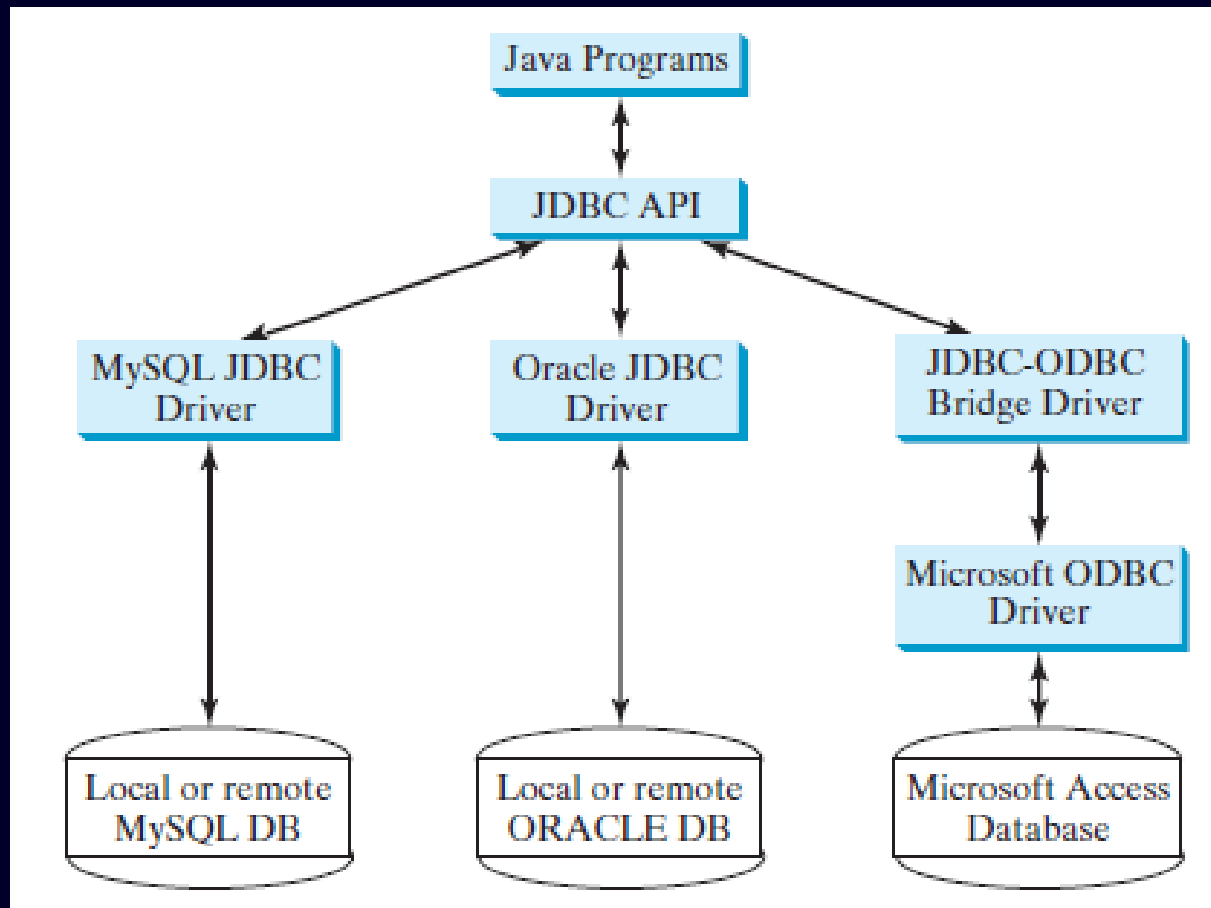


JDBC

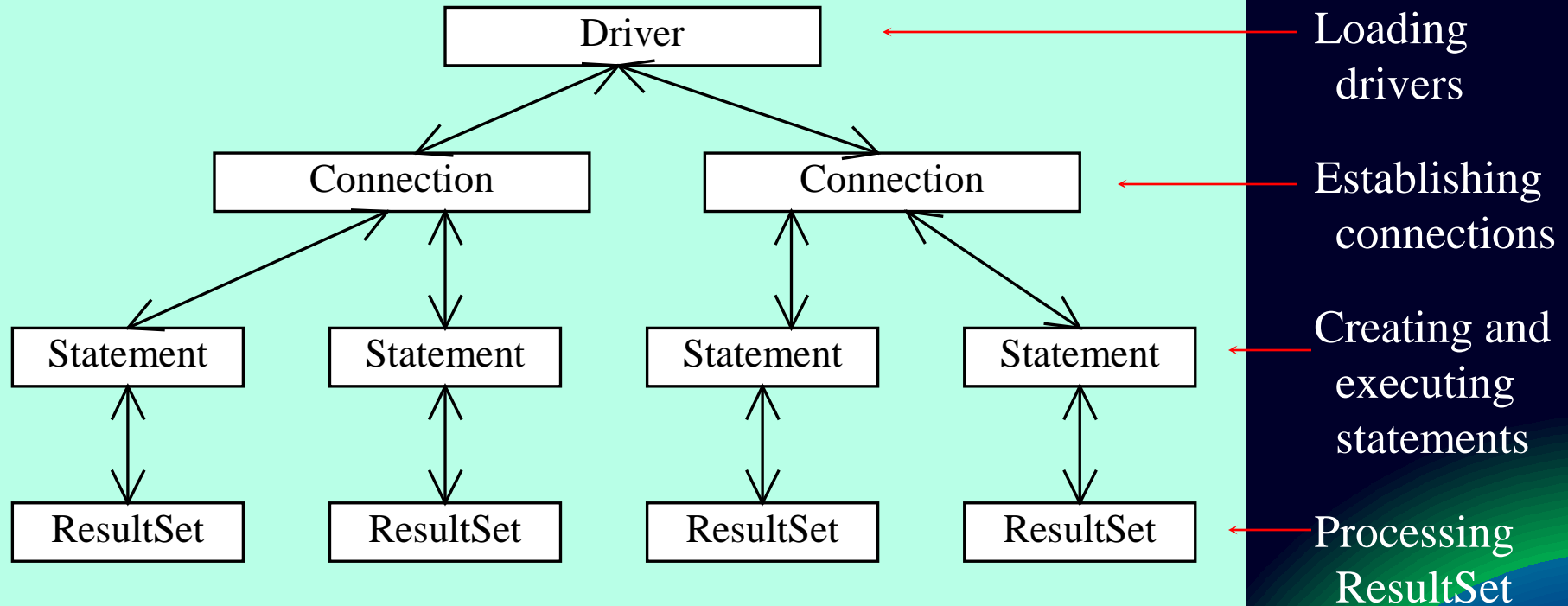
- *JDBC is the Java API for accessing relational databases.*
- Using the JDBC API, applications written in the Java programming language can execute SQL statements, retrieve results, present data in a user friendly interface, and propagate changes back to the database.

The Architecture of JDBC

The relationships between Java programs, JDBC API, JDBC drivers, and relational databases is shown below.



The JDBC Interfaces



JDBC classes enable Java programs to connect to the database, send SQL statements, and process results.

Developing JDBC Programs

Loading drivers

Establishing connections

Creating and executing statements

Processing ResultSet

Statement to load a driver:

```
Class.forName("JDBCDriverClass");
```

A driver is a class. For example:

Database	Driver Class	Source
Access	sun.jdbc.odbc.JdbcOdbcDriver	Already in JDK //Not Anymore
MySQL	com.mysql.jdbc.Driver	Website
Oracle	oracle.jdbc.driver.OracleDriver	Website

The JDBC-ODBC driver for Access is bundled in JDK.

MySQL driver class is in mysqljdbc.jar

Oracle driver class is in classes12.jar

To use the MySQL and Oracle drivers, you have to add mysqljdbc.jar and classes12.jar in the classpath using the following DOS command on Windows:

```
classpath=%classpath%;c:\book\mysqljdbc.jar;c:\book\classes12.jar
```

Developing JDBC Programs

Loading
drivers

Establishing
connections

Creating and
executing
statements

Processing
ResultSet

```
Connection connection = DriverManager.getConnection(databaseURL);
```

Database	URL Pattern
----------	-------------

Access	jdbc:odbc:dataSource
--------	----------------------

MySQL	jdbc:mysql://hostname/dbname
-------	------------------------------

Oracle	jdbc:oracle:thin:@hostname:port#:oracleDBSID
--------	----------------------------------------------

Examples:

For Access:

```
Connection connection = DriverManager.getConnection  
("jdbc:odbc:ExampleMDBDataSource");
```

See Supplement IV.D for
creating an ODBC data source

For MySQL:

```
Connection connection = DriverManager.getConnection  
("jdbc:mysql://localhost/test");
```

For Oracle:

```
Connection connection = DriverManager.getConnection  
("jdbc:oracle:thin:@liang.armstrong.edu:1521:orcl", "scott", "tiger");
```

Developing JDBC Programs

Loading
drivers

Establishing
connections

Creating and
executing
statements

Processing
ResultSet

Creating statement:

```
Statement statement = connection.createStatement();
```

Executing statement (for update, delete, insert):

```
statement.executeUpdate  
("create table Temp (col1 char(5), col2 char(5))");
```

Executing statement (for select):

```
// Select the columns from the Student table  
ResultSet resultSet = statement.executeQuery  
("select firstName, mi, lastName from Student where lastName "  
+ " = 'Smith'");
```

Developing JDBC Programs

Loading
drivers

Establishing
connections

Creating and
executing
statements

Processing
ResultSet

Executing statement (for select):

```
// Select the columns from the Student table
```

```
ResultSet resultSet = stmt.executeQuery
```

```
("select firstName, mi, lastName from Student where lastName " +  
+ " = 'Smith'");
```

Processing ResultSet (for select):

```
// Iterate through the result and print the student names
```

```
while (resultSet.next())
```

```
System.out.println(resultSet.getString(1) + " " + resultSet.getString(2)  
+ ". " + resultSet.getString(3));
```


Simple JDBC Program

```
import java.sql.*;

public class TestStudentsDB {
    public static void main(String[] args) throws SQLException {
        // Load the JDBC driver
        // Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        // System.out.println("Driver loaded");

        // Connect to a database
        //Connection connection = DriverManager.getConnection("jdbc:odbc:Students");

        Connection connection = DriverManager.getConnection("jdbc:ucanaccess://D:/Students.accdb");
        System.out.println("Database connected");

        // Create a statement
        Statement statement = connection.createStatement();

        // Execute a statement
        ResultSet resultSet = statement.executeQuery
            ("select firstName, lastName, Grade1, Grade2, Grade3 from StudentsTbl where lastName "
            + " = 'cruz'");

        // Iterate through the result and print the student names
        System.out.println("Name\tLast\tGrade1\tGrade2\tGrade3");

        while (resultSet.next()){
            System.out.println(resultSet.getString(1) + "\t" + resultSet.getString(2) + "\t" +
                resultSet.getDouble(3) + "\t" + resultSet.getDouble(4) + "\t" + resultSet.getDouble(5));
        }
        // Close the connection
        connection.close();
    }
}
```

Database connected

Name	Last	Grade1	Grade2	Grade3
rodolfo	cruz	100.0	90.0	98.0
Maria	cruz	78.0	89.0	90.0

Retrieving Database Metadata

- Database **metadata** is the information that describes database itself.
- **JDBC** provides the **DatabaseMetaData** interface for obtaining database wide information and the **ResultSetMetaData** interface for obtaining the information on the specific **ResultSet**.

DatabaseMetadata, cont.

- The **DatabaseMetaData** interface provides more than 100 methods for getting database metadata concerning the database as a whole.
- These methods can be divided into three groups: **for retrieving general information, for finding database capabilities, and for getting object descriptions.**

Metadata Example

```
import java.sql.*;

public class GetMetadata {
    public static void main(String[] args) throws SQLException {

        Connection connection = DriverManager.getConnection("jdbc:ucanaccess://D:/Teaching/Spring 2015/Java/Students.accdb");
        System.out.println("Database connected");

        DatabaseMetaData dbMetaData = connection.getMetaData();
        System.out.println("database URL: " + dbMetaData.getURL());
        System.out.println("database username: " +
            dbMetaData.getUserName());
        System.out.println("database product name: " +
            dbMetaData.getDatabaseProductName());
        System.out.println("database product version: " +
            dbMetaData.getDatabaseProductVersion());
        System.out.println("JDBC driver name: " +
            dbMetaData.getDriverName());
        System.out.println("JDBC driver version: " +
            dbMetaData.getDriverVersion());
        System.out.println("JDBC driver major version: " +
            dbMetaData.getDriverMajorVersion());
        System.out.println("JDBC driver minor version: " +
            dbMetaData.getDriverMinorVersion());
        System.out.println("Max number of connections: " +
            dbMetaData.getMaxConnections());
        System.out.println("MaxTableNameLength: " +
            dbMetaData.getMaxTableNameLength());
        System.out.println("MaxColumnsInTable: " +
            dbMetaData.getMaxColumnsInTable());

        // Close the connection
        connection.close();
    }
}
```

Sample Run

```
Database connected
database URL: jdbc:ucanaccess://D:/Teaching/Spring 2015/Java/Students.accdb
database username: Admin
database product name: Ucanaccess for access db(Jet) using hasqlldb
database product version: V2007 [VERSION_12]
JDBC driver name: Ucanaccess
JDBC driver version: 2.0.9.4
JDBC driver major version: 0
JDBC driver minor version: 0
Max number of connections: 0
MaxTableNameLength: 128
MaxColumnsInTable: 0
```

UCanAccess

<http://ucanaccess.sourceforge.net/site.html>