UNIVERSITY of HOUSTON

DEPARTMENT OF COMPUTER SCIENCE

COSC 3380 Spring 2024

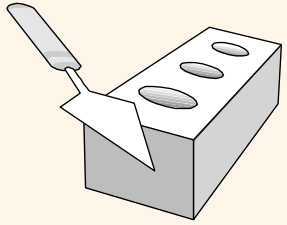
Database Systems

M & W 4:00 to 5:30 PM

Prof. Victoria Hilford

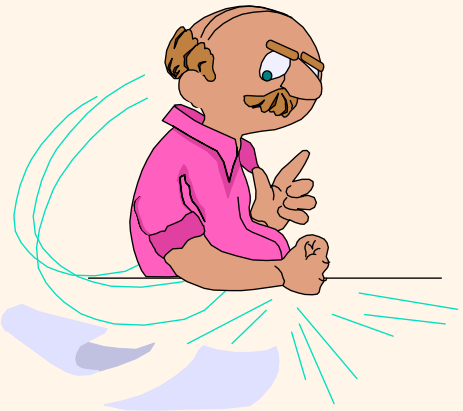
PLEASE TURN your webcam ON (must have)

NO CHATTING during LECTURE



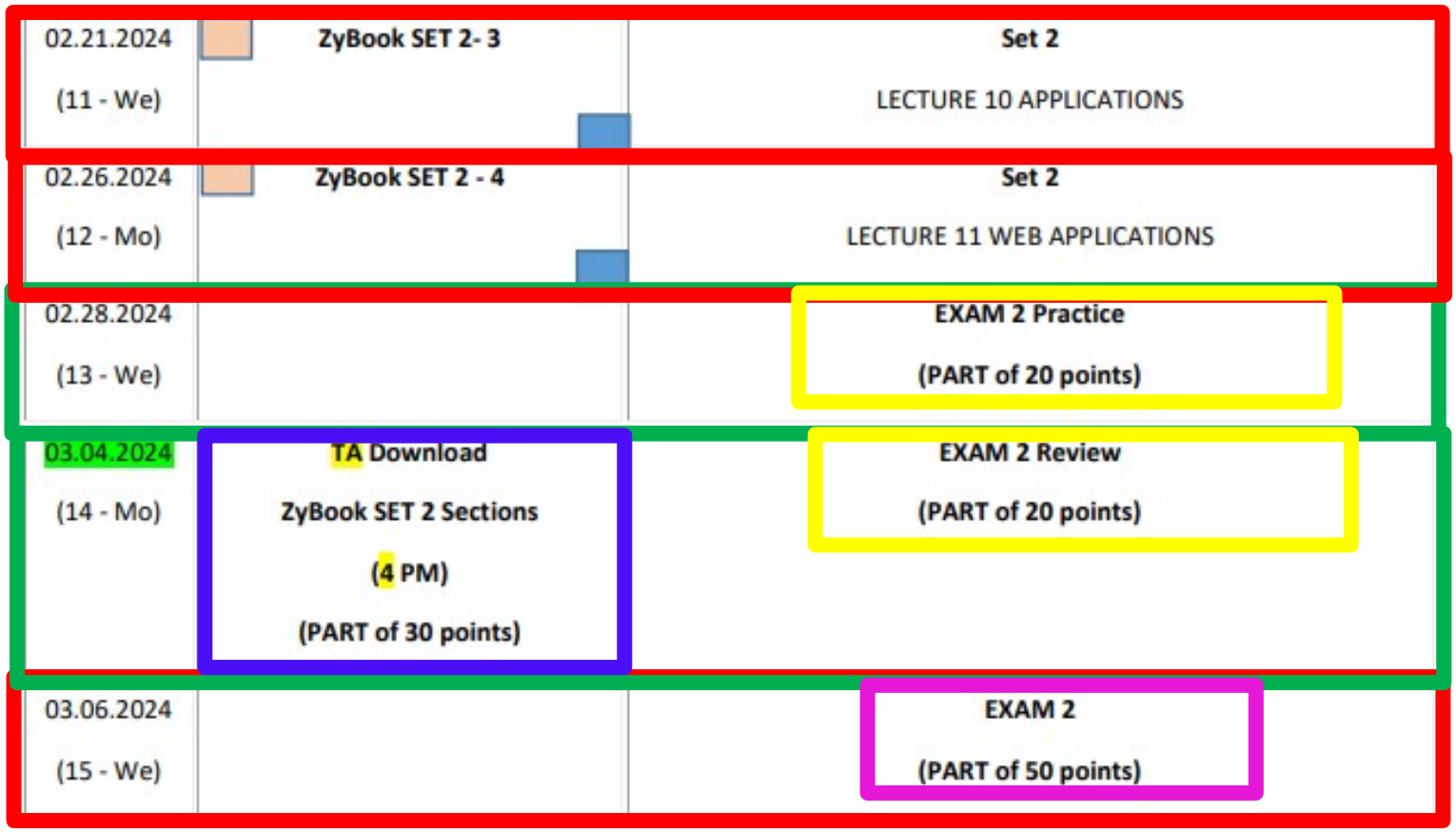
COSC 3380

4 to 5:30

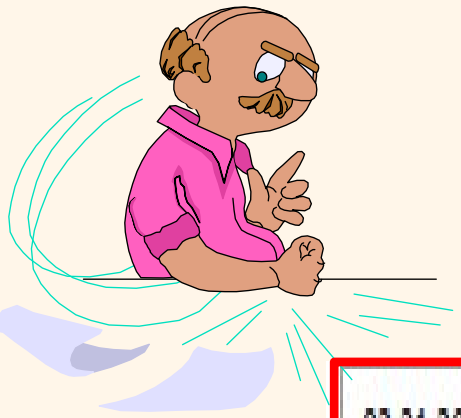


PLEASE
LOG IN
CANVAS

Please close all other windows.


02.21.2024
(11 - We)
ZyBook SET 2- 3
Set 2
LECTURE 10 APPLICATIONS
02.26.2024
(12 - Mo)
ZyBook SET 2 - 4
Set 2
LECTURE 11 WEB APPLICATIONS
02.28.2024
(13 - We)
EXAM 2 Practice
(PART of 20 points)
03.04.2024
(14 - Mo)
TA Download
ZyBook SET 2 Sections
(4 PM)
(PART of 30 points)
EXAM 2 Review
(PART of 20 points)
03.06.2024
(15 - We)
EXAM 2
(PART of 50 points)

COSC 3380



Class 11

02.21.2024	ZyBook SET 2-3	Set 2
(11 - We)		LECTURE 10 APPLICATIONS

From 4:00 to 4:07 PM – 5 minutes.

02.21.2024 (11 - We)	ZyBook SET 2-3	Set 2 LECTURE 10 APPLICATIONS

CLASS PARTICIPATION 20 points — 20% of Total

APPLICATIONS

Class 11 BEGIN PARTICIPATION
Not available until Feb 21 at 4:00pm | Due Feb 21 at 4:07pm | 100 pts

VH, publish

This is a synchronous online class.

Attendance is required.

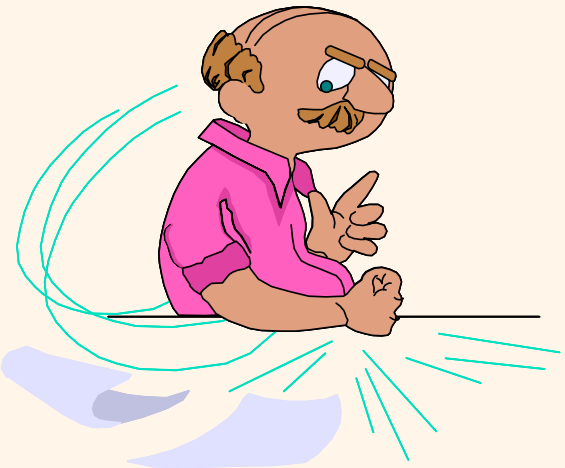
Recording or distribution of class materials is prohibited.

1. At the beginning of selected classes there is an **assessment in the first 10 minutes**. (**beige** BOX in the Detailed Syllabus)
2. At the end of selected classes there is an **assessment in the last 10 minutes**. (**blue** BOX in the Detailed Syllabus)
3. ZyBook sections will be downloaded and used for 30% of Total Score on the dates specified in the Detailed Syllabus.
4. EXAMS are in CANVAS. **No late EXAMS**.
5. I have to be **present in TEAMS in order to take any graded assignment assigned during that class.**

From 4:07 to 5:00 PM – 53 minutes.

Lecture 10

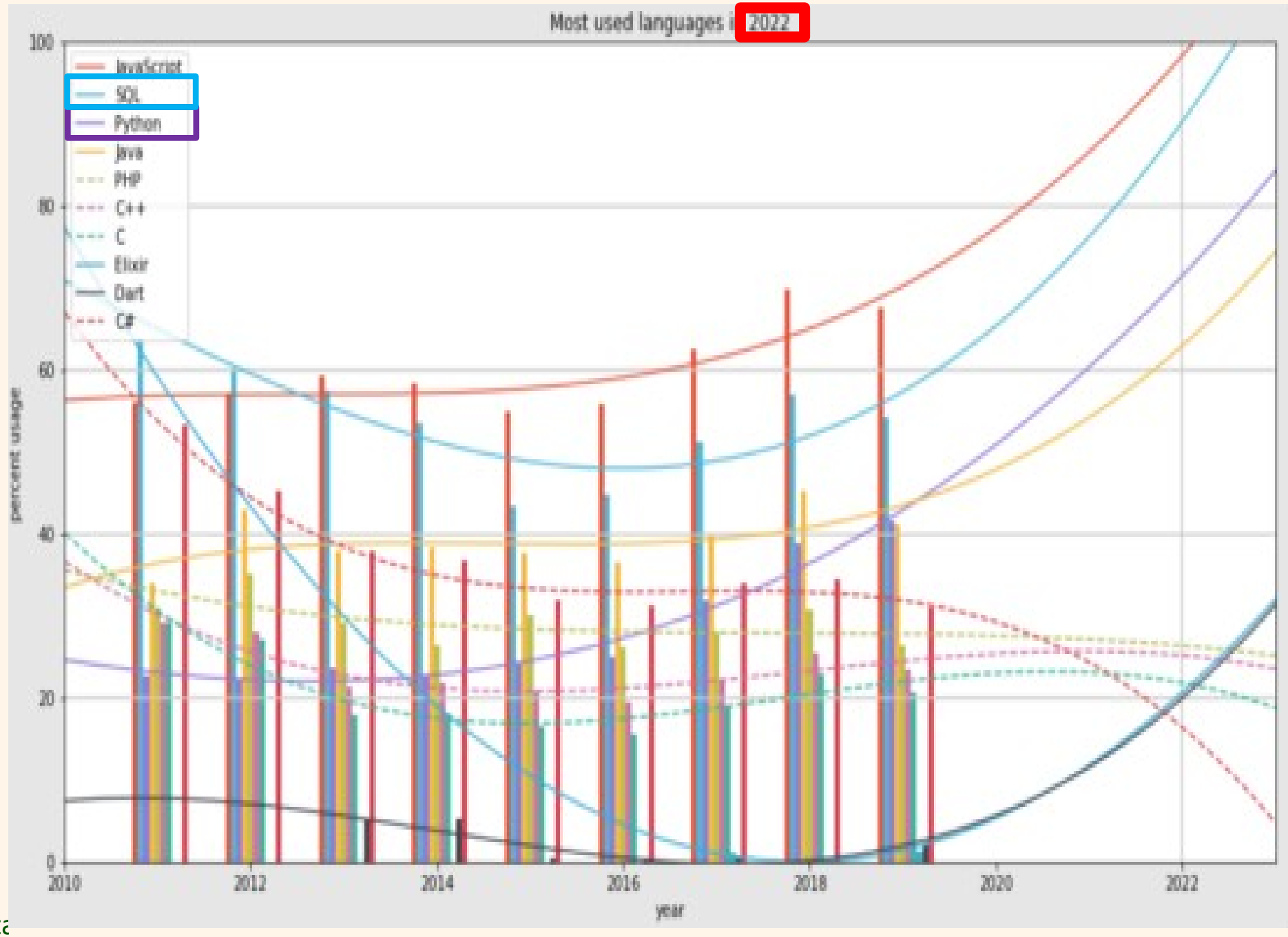
*Database **Application Development***



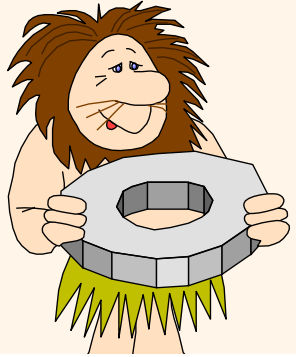

10. SET 2 - 3:APPLICATIONS
0%
0%
10.1 Programming languages
0%
10.2 Embedded SQL
0%
0%
10.3 Procedural SQL
0%
0%
10.4 Application programming interfaces
0%
10.5 Database programming with Python
0%
0%

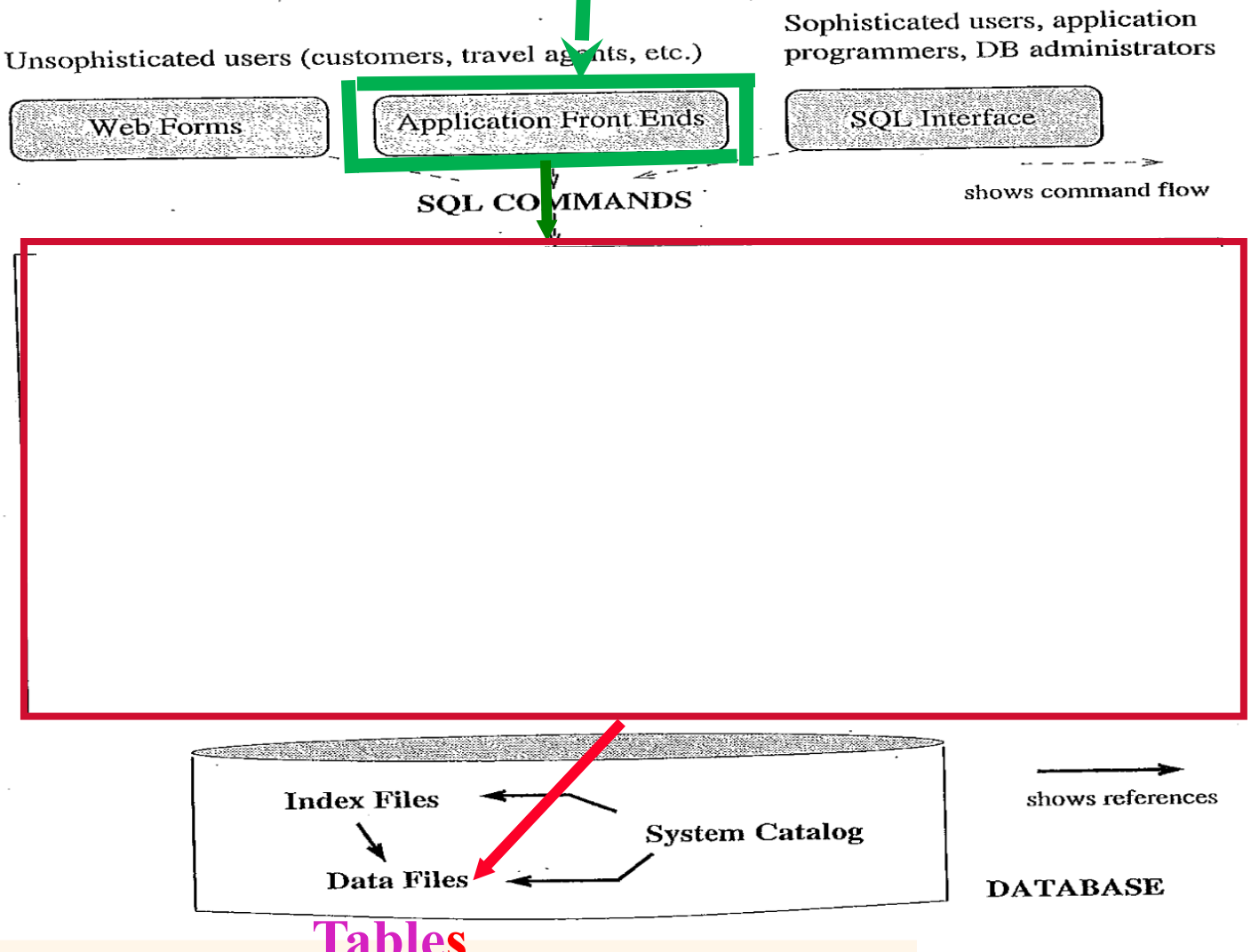
TOP 10 POPULAR APPS THAT USES PYTHON

Apps	Traffic [Similarweb]
Google	89 B
Instagram	6.5 B
Spotify	385 M
Pinterest	1 B
Uber	1 B
Netflix	2.5 B
Reddit	1.7 B
Dropbox	178 M
Instacart	28 M
Disqus	35 M


Most used languages in 2022
Javascript
SQL
Python
Java
PHP
C++
C
Elixir
Dart
C#
100
80
60
40
20
0
percent usage
2010
2012
2014
2016
2018
2020
2022
year

DBMS Application Software





10.1 Programming languages

Programming languages

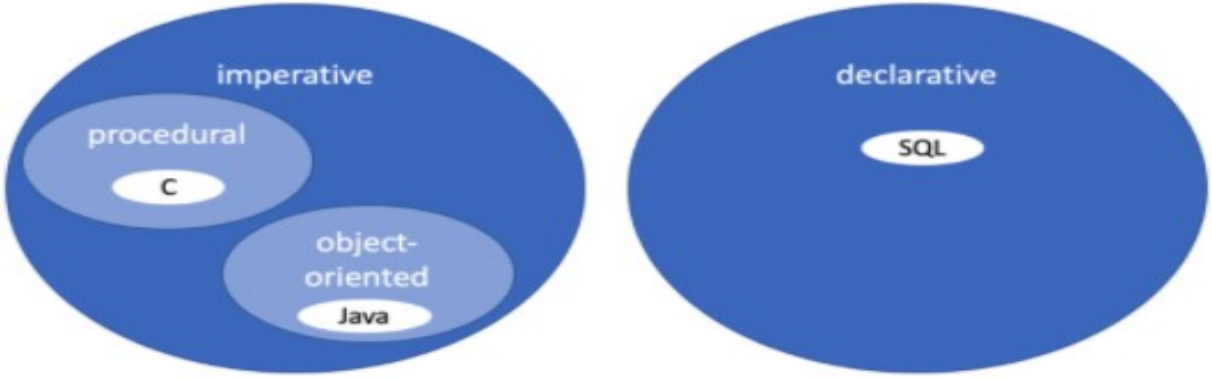
Programming languages fall into two broad categories, or paradigms: imperative and declarative.

Imperative languages contain control flow statements that determine the execution order of program steps. Control flow statements include loops for repeatedly executing code and conditionals for conditionally executing code. Two popular types of imperative languages are:

1. ***Procedural languages*** are composed of procedures, also called functions or subroutines. Most languages developed prior to 1990 are procedural. Ex: C and COBOL.
2. ***Object-oriented languages*** organize code into classes. A class combines variables and procedures into a single construct. Most languages developed since 1990 are object-oriented. Ex: Java, Python, and C++.

Declarative languages do not contain control flow statements. Each statement declares *what* result is desired, using logical expressions, rather than *how* the result is processed. Compilers for declarative languages are called ***optimizers***, since the compiler determines an optimal way to process each declarative statement. SQL is the leading example of a declarative language.

Figure 10.1.1: Programming languages.



Declarative language

```
SELECT AirlineName, FlightNumber
FROM Flight
WHERE AirportCode = 'JFK' AND DepartureTime > '12:00';
```

SQL

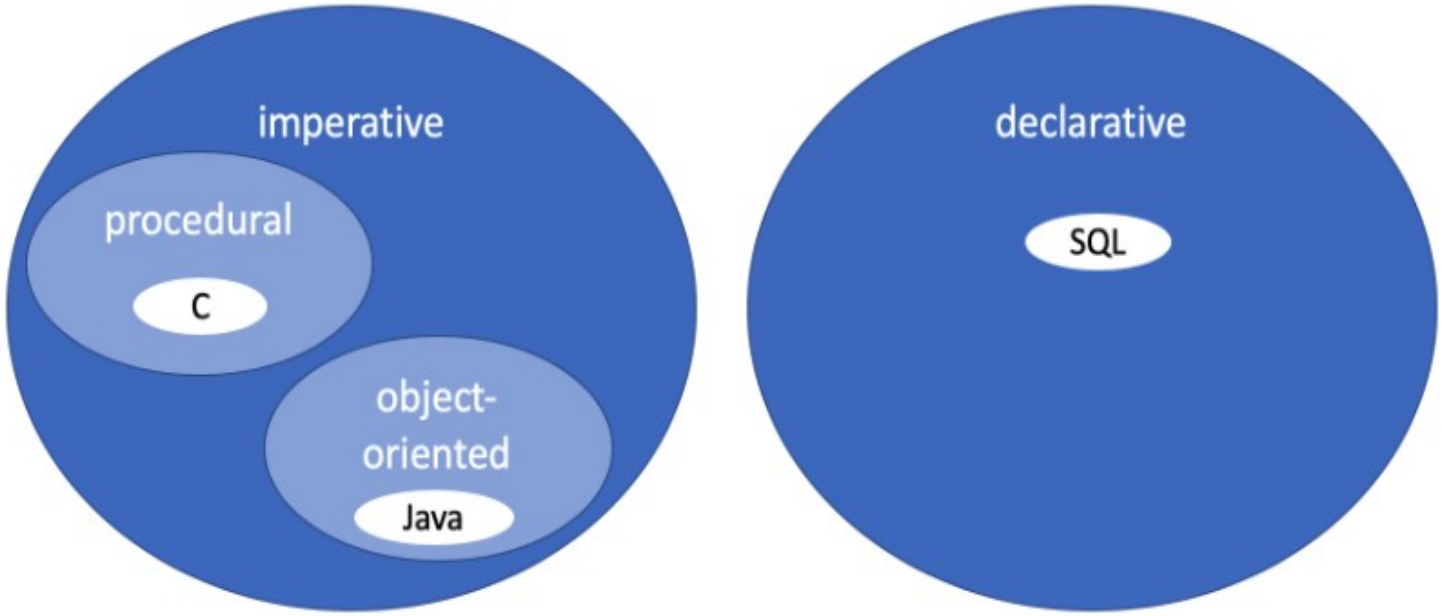
Procedural language

```
printf("%20s %12s \n", "AirlineName ", "FlightNumber");
for (int i = 1; i <= rowCount(flight); i++) {
    if (strcmp(getAirportCode(i), "JFK") == 0 && getDepartureTime(i) > 12)
        printf("%20s %12d \n", getAirlineName(i), getFlightNumber(i));
}
```

C

AirlineName	FlightNumber
American Airlines	1154
Air India	852
Lufthansa	920

Figure 10.1.1: Programming languages.



PARTICIPATION ACTIVITY | 10.1.3: Database programming.

1) _____ is the leading declarative language.

SQL

Correct

SQL

Although many database processing languages are declarative, SQL was backed by IBM when relational database first became popular. As a result, SQL became the most widely used declarative language.

?????

Concepts covered in this Lecture:

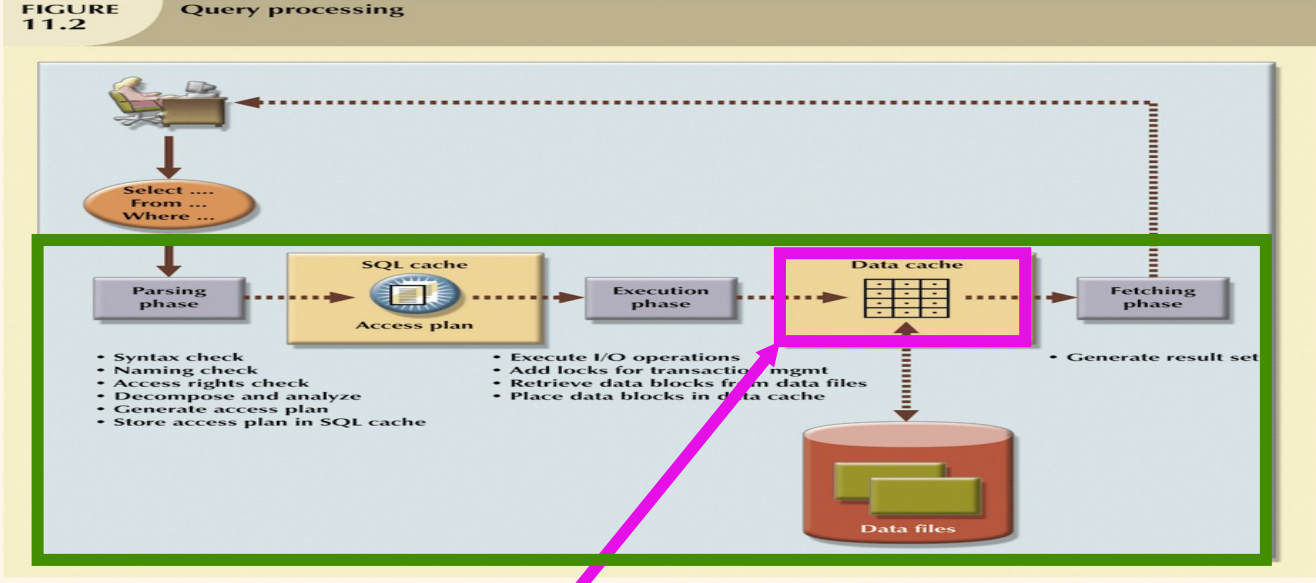
- **Cursors**
- **Stored procedures**

SQL in application code

- JDBC, ODBC (API)
- **SQL**J

- Embedded **SQL**
- Dynamic **SQL**

Cursors



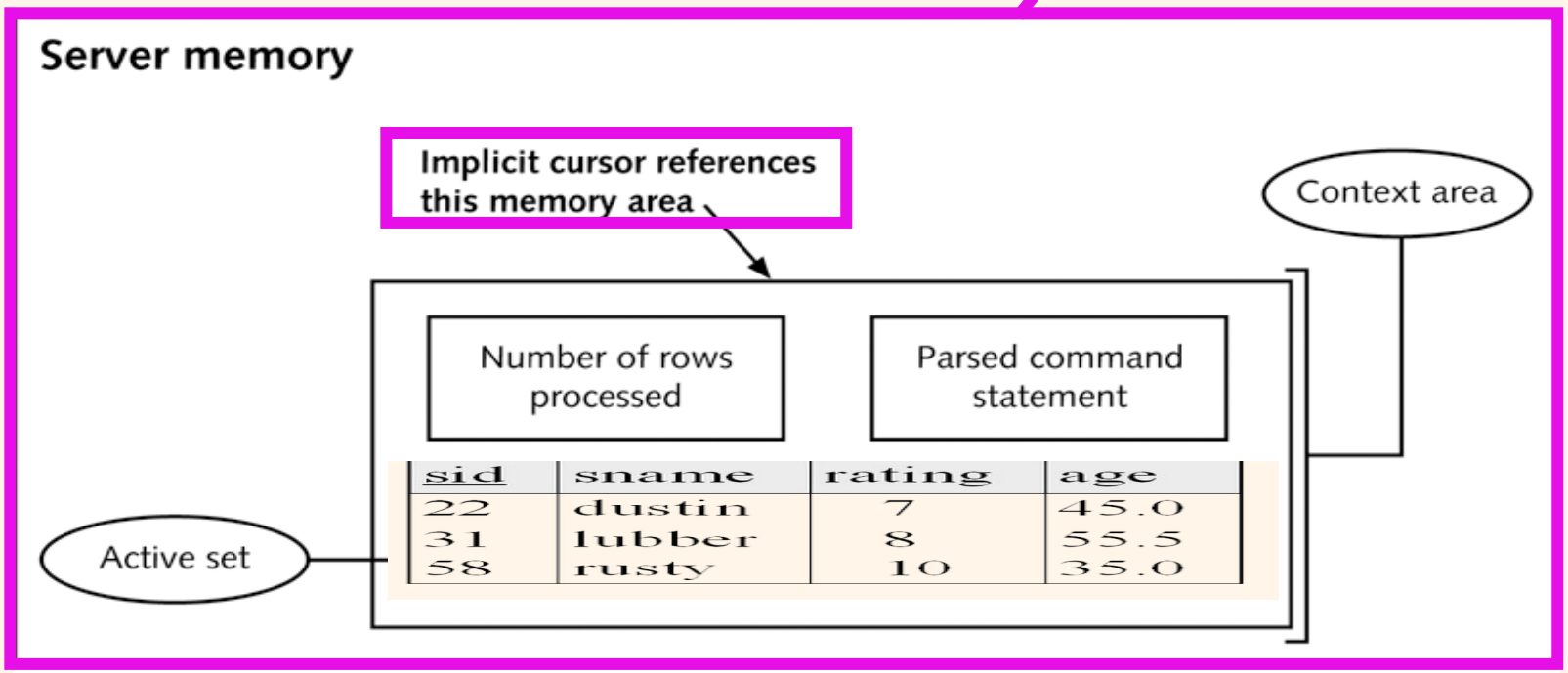


Figure 4-25 Implicit cursor

Cursors

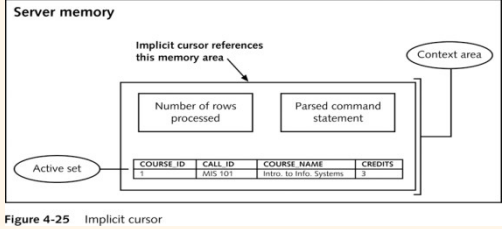


Figure 4-25 Implicit cursor

- Can declare a **cursor** on a **Relation** or **SQL query statement** (which generates a **Relation**).

- Can *open* a **cursor**, and repeatedly ***fetch*** a **tuple**, then *move* the **cursor**, until all **tuples** have been retrieved.

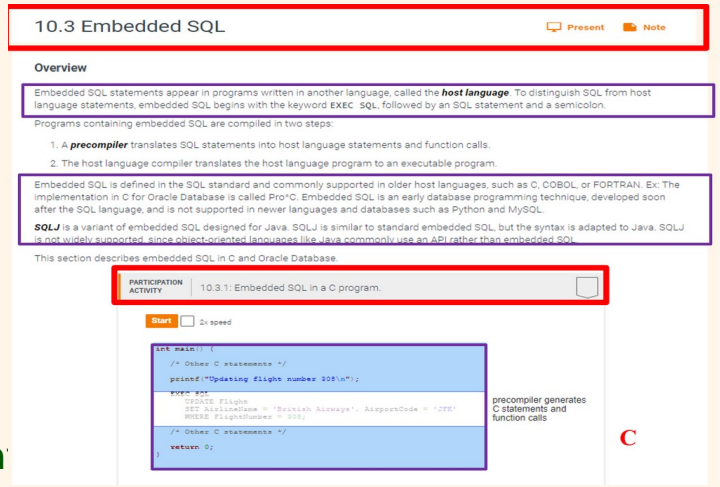
  - Can use a special clause, called ORDER BY, in queries that are accessed through a **cursor**, to control the order in which **tuples** are returned.
    - Fields in ORDER BY clause **must also appear** in SELECT clause.

  - The ORDER BY clause, which orders answer **tuples**, is *only* allowed in the context of a **cursor**.

Cursor sinfo *that gets names of sailors who've reserved a red boat, in alphabetical order*

in ***EMBEDDED SQL***

```
EXEC SQL DECLARE sinfo Cursor FOR
        SELECT  S.sname
        FROM  Sailors S, Boats B, Reserves R
        WHERE  S.sid=R.sid AND R.bid=B.bid AND
B.color='red'
        ORDER BY  S.sname
```



Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Sailors

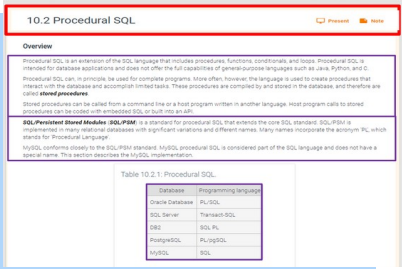
sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	green
103	Clipper	red
104	Marine	red

rke

```
CREATE PROCEDURE ChangeDepartureTime(IN airportIn CHAR(3))
BEGIN

    DECLARE flightNum INT;
    DECLARE airport CHAR(3);
    DECLARE departTime TIME;
    DECLARE finishedReading BOOL DEFAULT FALSE;

    DECLARE flightCursor CURSOR FOR
        SELECT FlightNumber, AirportCode, DepartureTime
        FROM Flight;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finishedReading = TRUE;

    OPEN flightCursor;
    FETCH FROM flightCursor INTO flightNum, airport, departTime;

    WHILE NOT finishedReading DO
        IF airport = airportIn THEN
            UPDATE Flight
            SET DepartureTime = ADDTIME(departTime, '1:00')
            WHERE FlightNumber = flightNum;
        ELSE
            UPDATE Flight
            SET DepartureTime = SUBTIME(departTime, '0:30')
            WHERE FlightNumber = flightNum;
        END IF;

        FETCH FROM flightCursor INTO flightNum, airport, departTime;
    END WHILE;

    CLOSE flightCursor;

END;
```


10.2 Procedural SQL

Overview

Procedural SQL is an extension of the SQL language that includes procedures, functions, conditionals, and loops. Procedural SQL is intended for database applications and does not offer the full capabilities of general-purpose languages such as Java, Python, and C.

Procedural SQL can, in principle, be used for complete programs. More often, however, the language is used to create procedures that interact with the database and accomplish limited tasks. These procedures are compiled by and stored in the database, and therefore are called *stored procedures*.

Stored procedures can be called from a command line or a host program written in another language. Host program calls to stored procedures can be coded with embedded SQL or built into an API.

SQL/Persistent Stored Modules (SQL/PSM) is a standard for procedural SQL that extends the core SQL standard. SQL/PSM is implemented in many relational databases with significant variations and different names. Many names incorporate the acronym 'PL,' which stands for 'Procedural Language.'

MySQL conforms closely to the SQL/PSM standard. MySQL procedural SQL is considered part of the SQL language and does not have a special name. This section describes the MySQL implementation.

Table 10.2.1: Procedural SQL.

Database	Programming language
Oracle Database	PL/SQL
SQL Server	Transact-SQL
DB2	SQL PL
PostgreSQL	PL/pgSQL
MySQL	SQL

Procedural SQL

TA, Jordan (A – L).

TA, Fernando (M – Z).

Please compare CANVAS vs. TEAMS Attendance.
Print screens of students in CANVAS but not in the TEAMS meeting.
(2.21.2024 Attendance X missing LastName.docx)

Concepts covered in this Lecture:

- **Cursors**
- **Stored procedures**

SQL in application code

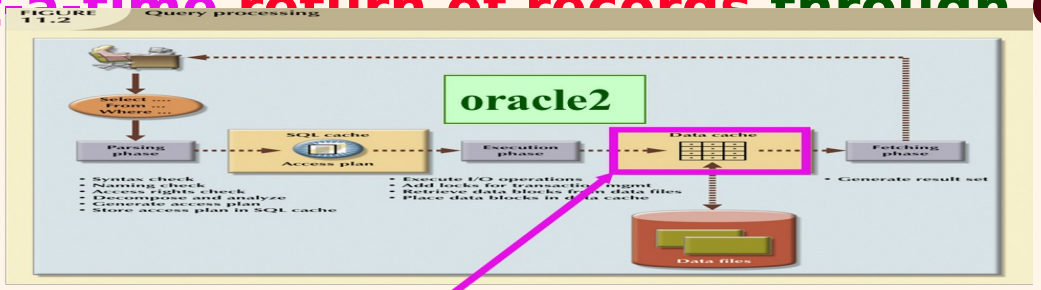
- JDBC, ODBC (API)
- **SQL**J

- Embedded **SQL**
- Dynamic **SQL**

Stored Procedures

- What is a **stored procedure**:
  - **Program executed through a single SQL statement**
  - **Executed in the process space of the DBMS server**

- Advantages:
  - **Can encapsulate application logic while staying "close" t the Data**
  - **Reuse of application logic by different users**
  - **Avoid tuple-at-a-time return of records through Cursors**



Stored Procedures

Stored Procedures:

Examples

```
CREATE PROCEDURE ShowNumReservations
    SELECT S.sid, S.sname, COUNT(*)
    FROM Sailors S, Reserves R
    WHERE S.sid = R.sid
    GROUP BY S.sid, S.sname
```

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Sailors

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

Stored procedures can have parameters:

- Three different modes: IN, OUT, INOUT

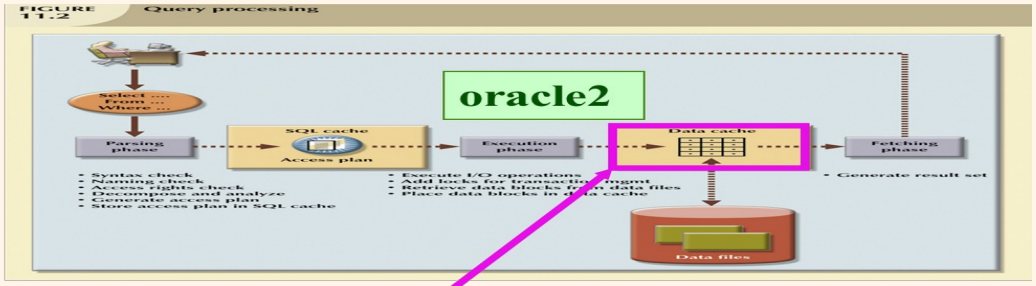
```
CREATE PROCEDURE IncreaseRating(
IN sailor_sid INTEGER, IN increase INTEGER)
        UPDATE Sailors
        SET rating = rating + increase
        WHERE sid = sailor_sid
```

Stored Procedures:

Examples

Stored procedure do not have to be written in **SQL**:

CREATE PROCEDURE TopSailors(IN num INTEGER)
LANGUAGE **JAVA**
EXTERNAL NAME "**file:///c:/storedProcs/rank.jar**"



Stored Procedures

Calling Stored Procedures

```
CREATE PROCEDURE IncreaseRating(
    IN sailor_sid INTEGER, IN increase INTEGER)
        UPDATE Sailors
        SET rating = rating + increase
        WHERE sid = sailor_sid
```

From Embedded **SQL**

	Paradigm gap	Syntax gap	Object-oriented language	Applications	Compile steps	Network traffic	Database-independent
Embedded SQL	High	High	Limited	General	2	High	No

```
EXEC SQL BEGIN DECLARE SECTION
int sid;
int increase;
EXEC SQL END DECLARE SECTION

// now increase the rating of this sailor

EXEC CALL IncreaseRating(:sid, :increase);
```

Calling Stored Procedures

JDBC:

```
CallableStatement cstmt= con.prepareCall("{CALL
    ShowSailors}");
ResultSet rs = cstmt.executeQuery();
while (rs.next())
{
   ...
}
```

	Paradigm gap	Syntax gap	Object-oriented language	Applications	Compile steps	Network traffic	Database-independent
API	Moderate	Low	Yes	General	1	Variable	Yes

SQLJ:

```
#sql iterator ShowSailors(...);
ShowSailors showsailors;
#sql showsailors={CALL ShowSailors};
while (showsailors.next())
{
   ...
}
```

Table 10.1.1: Advantages and disadvantages.

	Paradigm gap	Syntax gap	Object-oriented language	Applications	Compile steps	Network traffic	Database-independent
Procedural SQL	Moderate	Low	Limited	Database processing	1	Low	No

10.2 Procedural SQL

Overview

Procedural SQL is an extension of the SQL language that includes procedures, functions, conditionals, and loops. Procedural SQL is intended for database applications and does not offer the full capabilities of general-purpose languages such as Java, Python, and C.

Procedural SQL can, in principle, be used for complete programs. More often, however, the language is used to create procedures that interact with the database and accomplish limited tasks. These procedures are compiled by and stored in the database, and therefore are called ***stored procedures***.

Stored procedures can be called from a command line or a host program written in another language. Host program calls to stored procedures can be coded with embedded SQL or built into an API.

SQL/Persistent Stored Modules (***SQL/PSM***) is a standard for procedural SQL that extends the core SQL standard. SQL/PSM is implemented in many relational databases with significant variations and different names. Many names incorporate the acronym 'PL', which stands for 'Procedural Language'.

MySQL conforms closely to the SQL/PSM standard. MySQL procedural SQL is considered part of the SQL language and does not have a special name. This section describes the MySQL implementation.

Table 10.2.1: Procedural SQL.

Database	Programming language
Oracle Database	PL/SQL
SQL Server	Transact-SQL
DB2	SQL PL
PostgreSQL	PL/pgSQL
MySQL	SQL

CHALLENGE ACTIVITY | 10.2.1: Procedural SQL.

```
CREATE PROCEDURE AddFlights(IN startTime TIME, IN endTime TIME)
BEGIN
   DECLARE departTime TIME DEFAULT startTime;
   DECLARE flightNum INT DEFAULT 540;

   WHILE departTime <= endTime DO

      INSERT INTO Flight
      VALUES (flightNum, 'Lufthansa Airlines', departTime, 'FRA');

      SET flightNum = flightNum + 100;

      IF departTime < '12:00' THEN
         SET departTime = ADDTIME(departTime, '00:30');
      ELSE
         SET departTime = ADDTIME(departTime, '1:00');
      END IF;

   END WHILE;
END;
```

Hey, what's another programming language?!

CHALLENGE ACTIVITY | 10.2.1: Procedural SQL.

```
CREATE TABLE Seat (
    RoomName VARCHAR(40),
    SeatNumber INT,
    PRIMARY KEY (RoomName, SeatNumber)
);
```

Fill in the keywords to create a stored procedure that adds entries to the Seat table.

```
CREATE PROCEDURE AddSeats(IN roomName VARCHAR(40), IN howMany INT)
BEGIN
    DECLARE n INT DEFAUT 1;

    WHILE n <= howMany DO
        INSERT INT Seat VALUES (roomName, n);
        SET n = n + 1;
    END WHILE;
END;
```

1

TA time (Jordan) – 5 minutes
(CA 10.2.1 Step 1 – Procedural SQL)

CHALLENGE ACTIVITY | 10.2.1: Procedural SQL.

```
CREATE TABLE Seat (
   RoomName VARCHAR(40),
   SeatNumber INT,
   PRIMARY KEY (RoomName, SeatNumber)
);
```

Fill in the keywords to create a stored procedure that adds entries to the Seat table.

```
___(A)___ PROCEDURE AddSeats(IN roomName VARCHAR(40), ___(B)___ howMany ___(C)___)
BEGIN
   DECLARE n INT DEFAULT 1;

   WHILE n <= howMany DO
      INSERT INTO Seat VALUES (roomName, n);
      ___(D)___ n = n + 1;
   END ___(E)___;
___(F)___;
```

(A)	Ex: KEYWORD	(D)	
(B)		(E)	
(C)		(F)	

1

COSC 3380 23578 M & W 4 to 5:30 PM LECTURE (in TEAMS from 3:50 to 5:15 PM)

44:49

Take control | Pop out | Chat | People 110 | Raise | View | Rooms | Apps | More | Camera | Mic | Share | Leave

Yu, Jordan T | Rahman, Z... | View all

Section 10.2 - COSC 3380: D | Section 10.6 - COSC 3380: D | PreparedStatement (Java Pla | Inbox (11,313) - jyu32000@g

learn.zybooks.com/zybook/UHCOSC3380HilfordSpring2024/chapter/10/section/2

Relaunch to update

Inbox (1,324) | Google Docs | GPT-4 | 2024SP COSC639... | NeRFrac: Neural R... | Fantasy Basketball... | AccessUH | The Perfect Practi... | ChatGPT | All Bookmarks

zyBooks My library > COSC 3380: Database Systems home > 10.2: Procedural SQL

zyBooks catalog | Help/FAQ | Jordan Yu

```
__(A)__ PROCEDURE AddSeats(IN roomName VARCHAR(40), __(B)__ howMany __(C)__
__(D)__
   DECLARE n INT DEFAULT 1;

   WHILE n <= howMany DO
      INSERT INTO Seat VALUES (roomName, n);
      __(E)__ n = n + 1;
   END __(F)__;
END;
```

(A)	CREATE	(D)	BEGIN
(B)	IN	(E)	SET
(C)	INT	(F)	WHILE

1 2

Check | Next

✔ Expected:

(A) CREATE
(B) IN
(C) INT
(D) BEGIN
(E) SET
(F) WHILE

(A) CREATE PROCEDURE begins the definition of a stored procedure.
(B) IN howMany INT declares howMany as an input parameter.
(C) IN howMany INT declares the type of howMany to be INT.
(D) BEGIN begins a compound statement.
(E) SET n = n + 1; sets the value of n to n + 1.
(F) END WHILE ends a WHILE statement.

View solution (Instructors only)

Feedback?

Yu, Jordan T

Participants

Invite someone or dial a number

Share invite

In this meeting (110) | Mute all

- Hilford, Victoria — Organizer
- RA Adhikari, Rohit
- AA Akram, Ali
- BA Akukwe, Benetta O
- SA Altaf, Sameer
- SA Alvarez, Stephanie
- OA Anayor-Achu, Ogochukwu E
- HA Avci, Hatice Kubra
- RA Aysola, Riya
- AB Bahl, Anish
- SB Banza, Sean Paolo B
- HB Bui, Hieu
- Burger, Jake
- VC Carrillo-Zepeda, Victor E

Stored procedure calls

Stored procedures are part of MySQL procedural SQL. Stored procedures are saved in the database and often called from general-purpose languages like Python.

Stored procedures are called with the ***cursor.callproc()*** method, which has two parameters:

- The name of the stored procedure
- An input tuple containing a value for each stored procedure parameter

The method returns an output tuple, also containing one value for each stored procedure parameter.

Stored procedure parameters are designated IN, OUT, or INOUT, indicating the parameter is for input, output, or both. For an OUT parameter, the corresponding value of the input tuple is ignored. For an IN parameter, the corresponding value of the output tuple is the input value.

PARTICIPATION ACTIVITY 10.5.9: Calling a stored procedure to get flight count.

Procedural SQL

```
CREATE PROCEDURE FlightCount(IN airline VARCHAR(20), OUT quantity INT)
   SELECT COUNT(*)
   INTO quantity
   FROM Flight
   WHERE AirlineName = airline;
```

Memory

China Airlines	flightData
0	
China Airlines	result
23	

Python

```
flightCursor = reservationConnection.cursor()
flightData = ('China Airlines', 0)
result = flightCursor.callproc('FlightCount', flightData)
print(result[1])
flightCursor.close()
```

Python API!

SQL/PSM

Structured Query Language/Persistent Stored Modules

Most **DBMS**s allow users to write **stored procedures** in a simple,

general-purpose language (close to **SQL**) ☾ **SQL/PSM** standard

(ORACLE's **version** is called **PL/SQL)**

(Microsoft 's **version** is called **T - SQL**)

Declare a stored procedure:

CREATE PROCEDURE name(p1, p2, …, pn)

local variable declarations

procedure code;

Declare a function:

CREATE FUNCTION name (p1, …, pn) **RETURNS** **sqlDataType**

Main SQL/PSM Constructs

```
CREATE FUNCTION rateSailor (IN sailorId INTEGER) RETURNS
    INTEGER
DECLARE rating     INTEGER
DECLARE numRes INTEGER

SET numRes = (SELECT COUNT(*)
                     FROM Reserves R
                    WHERE R.sid = sailorId)

IF (numRes > 10) THEN
    rating =1;
ELSE
    rating = 0;
END IF;

RETURN rating;
```

Main SQL/PSM Constructs

SQL programming language!

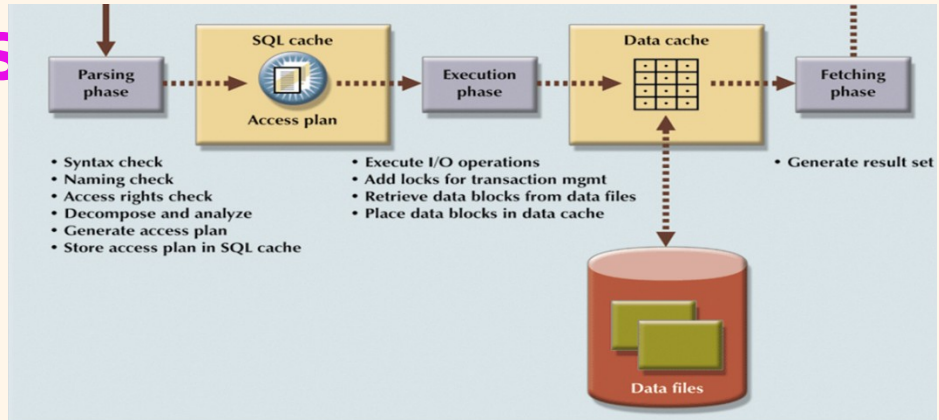
- Local variables (**DECLARE**)
- **RETURN** values for FUNCTION
- Assign variables with **SET**
- Branches and loops:
  - **IF** (condition) **THEN** statements; **ELSEIF** (condition) statements; … **ELSE** statements; **END IF**;
  - **LOOP** statements; **END LOOP**
- **Queries** can be parts of expressions
- Can use **Cursors** naturally

```
CREATE FUNCTION rateSailor (IN sailorId INTEGER) RETURNS INTEGER
DECLARE rating   INTEGER
DECLARE numRes INTEGER

SET numRes = (SELECT COUNT(*)
              FROM Reserves R
              WHERE R.sid = sailorId)

IF (numRes > 10) THEN rating =1;
ELSE rating = 0;
END IF;

RETURN rating;
```

SQL in Application Code

- **SQL** commands can be called from within a host language (e.g., **C**, **C++, Java, Python, etc.**) **program**.
  - **SQL** statements can refer to host variables (including special variables used to return status).
  - Must include a statement to *connect* to the right database.

- Two main integration approaches:
  - Embed **SQL** in the host language (**Embedded SQL**, **SQLJ**)
  - Create special API to call **S** **ODBC**)



Concepts covered in this Lecture:

- Cursors
- Stored procedures

SQL **in application code**

- JDBC, ODBCB (API)
- **SQL**J

- Embedded **SQL**
- Dynamic **SQL**

Table 10.1.1: Advantages and disadvantages.

	Paradigm gap	Syntax gap	Object-oriented language	Applications	Compile steps	Network traffic	Database-independent
API	Moderate	Low	Yes	General	1	Variable	Yes

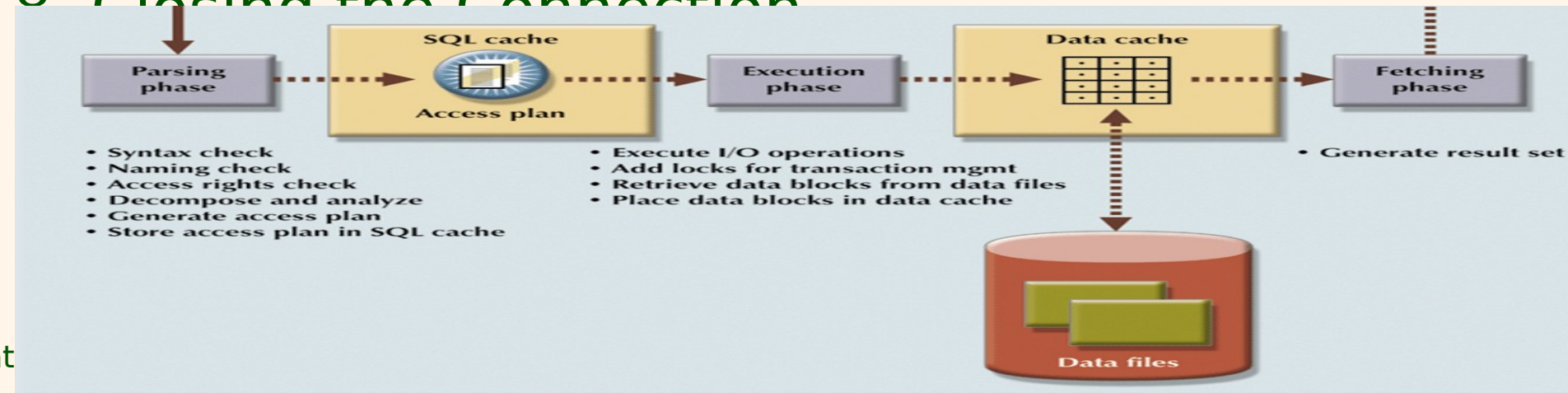
Java & JDBC **C# & ODBC** **EXCEL & ODBC**

Python & MysqlConnection

The JDBC Steps

	Paradigm gap	Syntax gap	Object-oriented language	Applications	Compile steps	Network traffic	Database-independent
API	Moderate	Low	Yes	General	1	Variable	Yes

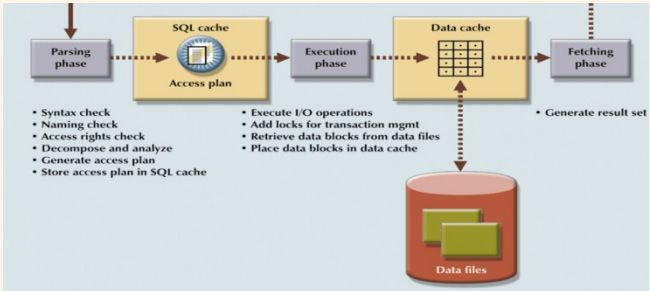
1. Importing Packages
2. Registering the JDBC Drivers
3. Opening a Connection to a Database
4. Creating a Statement **Object**
5. Executing a Query and Returning a Result Set **Object**
6. Processing the Result Set
7. Closing the Result Set and Statement **Objects**
8. Closing the Connection



JDBC

```
// Purpose: Basic selection using prepared statement
//

    //1. Import packages
    import java.sql.*;  //JDBC packages
    import java.math.*;
    import java.io.*;
    import oracle.jdbc.driver.*;

class Example{

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

// 2. Load Oracle driver
    DriverManager.registerDriver (new oracle.jdbc.driver.OracleDriver());
//Prompt user for username and password
    String user;
    String password;

    user     = readEntry("username: ");
    password = readEntry("password: ");

// 3. Connect to the local database
    Connection conn = DriverManager.getConnection ("jdbc:oracle:thin:@aardvark:1526:teach", user, password);
// Query the hotels table for resort = 'palma nova'
// Please notice the essential trim

// 4. Create a statement object
    PreparedStatement pstmt = conn.prepareStatement ("SELECT hotelname, rating FROM hotels WHERE trim(resort) = ?");
    pstmt.setString(1, "palma nova");

// 5. Execute Query
    ResultSet rset = pstmt.executeQuery ();

// 6. Process query results
    while (rset.next ())
       System.out.println (rset.getString (1)+" "+ rset.getString(2));


// 7. close the result set, statement, and 8. the connection
    rset.close();
    pstmt.close();
    conn.close();
}
```

	Paradigm gap	Syntax gap	Object-oriented language	Applications	Compile steps	Network traffic	Database-independent
API	Moderate	Low	Yes	General	1	Variable	Yes



10.4 Application programming interfaces

Overview

An application programming interface (API) specifies the interaction between an application and a computer service, such as a database, email, or web service. If the application is written in a procedural language like C, the API specifies procedure calls. If the application is written in an object-oriented language like Java, the API specifies classes.

Hundreds of APIs are available, connecting major programming languages to commonly used services. Dozens of APIs are available for relational databases and other data sources. Each programming language supports different APIs for different services. Ex: Java supports JMS for message services, JSAPI for speech synthesis and recognition services, JavaMail for email services, and many others. Conversely, each service may have different APIs for different programming languages.

Leading database APIs include:

- ***SQL/Call Language Interface*** (***SQL/CLI***), an extension of the SQL standard
- ***ODBC*** supports many programming languages and data sources. ODBC was developed in parallel with SQL/CLI and conforms closely to the standard.
- ***JDBC*** for Java applications
- ***DB-API*** for Python applications
- ***ADO.NET*** for .NET applications. .NET is a Microsoft environment, and C# is the most popular .NET language.
- ***PDO*** for PHP applications. PHP is a widely used programming language for building dynamic websites.

MySQL connectors implement leading APIs between the MySQL database and major programming languages.

Table 10.4.1: Leading database APIs.

Acronym	Derivation	Original Sponsor	Initial release	Application language	MySQL Connector
ODBC	Open Database Connectivity	Microsoft	1992	Numerous	Connector/ODBC
JDBC	Java Database Connectivity	Sun Microsystems (now Oracle)	1997	Java	Connector/J
DB-API	Database API	Python Software Foundation	1996	Python	Connector/Python
ADO.NET	ActiveX Data Objects	Microsoft	2002	.NET languages	Connector/NET
PDO	PHP Data Objects	The PHP Group	2005	PHP	none

10.4 Application programming interfaces

- An ***application programming interface***, or ***API***, is a library of procedures or classes. The library links an application programming language to a computer service, such as a database, email, or web service. The procedure or class declarations are written in the application programming language. Ex: JDBC is a library of Java classes that access relational databases.

PARTICIPATION ACTIVITY 10.1.4: API example using C# and ODBC.

```
string reservationConnString =
        "DRIVER={MySQL ODBC 3.51 Driver};" +
        "SERVER=localhost;" +
        "DATABASE=Reservation;" +
        "UID=samsnead;" +
        "PASSWORD=e&8nH!";
OdbcConnection reservationConnection = new OdbcConnection(reservationConnString);
reservationConnection.Open();

OdbcCommand listFlights = new OdbcCommand("SELECT * FROM Flight", reservationConnection);

OdbcDataReader flightReader;
flightReader = listFlights.ExecuteReader();

while (flightReader.Read()) {
   Console.WriteLine("Flight Number: " + flightReader.GetInt32(0));
   Console.WriteLine("Airline Name: " + flightReader.GetString(1));
}

flightReader.Close();
```

```
Flight Number: 208
Airline Name: United Airlines
Flight Number: 920
Airline Name: Lufthansa
```

C# & ODBC

Procedural SQL

```
CREATE PROCEDURE FlightCount(IN airline VARCHAR(20), OUT quantity INT)
  SELECT COUNT(*)
  INTO quantity
  FROM Flight
  WHERE AirlineName = airline;
```

Java

```
import java.sql.*;
import java.util.Scanner;

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

      Connection reservation = DriverManager.getConnection("jdbc:mysql://localhost/Reservation",
                                                           "samsnead", "b98$xm");
      CallableStatement flightCall = reservation.prepareCall("{ CALL FlightCount(?, ?) }");
      System.out.print("Enter airline name: ");
      Scanner scnr = new Scanner(System.in);
      String airlineName = scnr.nextLine();

      flightCall.setString("Airline", airlineName);
      flightCall.registerOutParameter("Quantity", Types.INTEGER);
      flightCall.executeQuery();
      int total = flightCall.getInt("Quantity");

      System.out.println("Airline: " + airlineName + "\nTotal flights: " + total);
      reservation.close();
   }
}
```

CHALLENGE ACTIVITY | 10.6.1: Database programming with Java.

Java

```
import java.sql.*;
import java.util.Scanner;

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

      Connection reservation = DriverManager.getConnection("jdbc:mysql://localhost/Reservation",
                                                           "samsnead", "b98$xm");
      CallableStatement flightCall = reservation.prepareCall("{ CALL FlightCount(?, ?) }");
      System.out.print("Enter airline name: ");
      Scanner scnr = new Scanner(System.in);
      String airlineName = scnr.nextLine();

      flightCall.setString("Airline", airlineName);
      flightCall.registerOutParameter("Quantity", Types.INTEGER);
      flightCall.executeQuery();
      int total = flightCall.getInt("Quantity");

      System.out.println("Airline: " + airlineName + "\nTotal flights: " + total);
      reservation.close();
   }
}
```

TA time (Jordan) – 5 minutes

(CA 10.6.1 –Database programming with Java)

CHALLENGE ACTIVITY | 10.6.1: Database programming with Java.

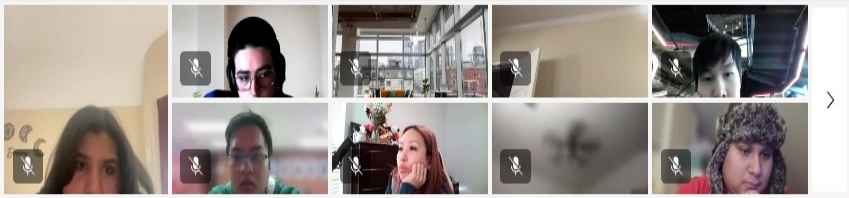
Java

```
import java.sql.*;
import java.util.Scanner;

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

      Connection reservation = DriverManager.getConnection("jdbc:mysql://localhost/Reservation",
                                                           "samsnead", "b98$xm");
      CallableStatement flightCall = reservation.prepareCall("{ CALL FlightCount(?, ?) }");
      System.out.print("Enter airline name: ");
      Scanner scnr = new Scanner(System.in);
      String airlineName = scnr.nextLine();

      flightCall.setString("Airline", airlineName);
      flightCall.registerOutParameter("Quantity", Types.INTEGER);
      flightCall.executeQuery();
      int total = flightCall.getInt("Quantity");

      System.out.println("Airline: " + airlineName + "\nTotal flights: " + total);
      reservation.close();
   }
}
```

COSC 3380 23578 M & W 4 to 5:30 PM LECTURE (in TEAMS from 3:50 to 5:15 PM)

01:11:04

Take control | Pop out | Chat | People 112 | Raise | View | Rooms | Apps | More | Camera | Mic | Share | Leave



Yu, Jordan T | Rahman, Z... | View all

Participants

Invite someone or dial a number

Share invite

In this meeting (112) Mute all

- Hilford, Victoria — Organizer
- RA Adhikari, Rohit
- MA Ahmed, Mohamed A
- AA Akram, Ali
- BA Akukwe, Benetta O
- SA Altaf, Sameer
- SA Alvarez, Stephanie
- OA Anayor-Achu, Ogochukwu E
- HA Avci, Hatice Kubra
- RA Aysola, Riya
- AB Bahl, Anish
- SB Banza, Sean Paolo B
- HB Bui, Hieu
- Burger, Jake

Section 10.2 - COSC 3380: D | Section 10.6 - COSC 3380: D | PreparedStatement (Java Pla

learn.zybooks.com/zybook/UHCOSC3380HilfordSpring2024/chapter/10/section/6

Relaunch to update

Inbox (1,324) | Google Docs | GPT-4 | 2024SP COSC639... | NeRFrac: Neural R... | Fantasy Basketball... | AccessUH | The Perfect Practi... | ChatGPT | All Bookmarks

zyBooks My library > COSC 3380: Database Systems home > 10.6: Database programming with Java

zyBooks catalog | Help/FAQ | Jordan Yu

the missing keywords.

```
Connection roomConn = null;
roomConn = DriverManager._____(A)_____(
    "jdbc:mysql://127.0.0.1/reservation?"
        + "user=samsnead&password=*jksi72$");
String roomQuery = "SELECT RoomNumber FROM Room WHERE FloorNumber = ?";
PreparedStatement roomStatement = roomConn._____(B)_____(roomQuery);
roomStatement._____(C)_____(1, 3);
ResultSet resultSet = roomStatement._____(D)_____();
while (resultSet.next()) {
    System.out.println("Found room " + resultSet.getInt("_____(E)_____")
        + " located on the 3rd floor.");
}
resultSet._____(F)_____();
_____(G)_____.close();
roomConn.close();
```

(A) getConnection
(B) prepareStatement
(C) SetInt
(D) executeQuery
(E) RoomNumber
(F) close
(G) roomStatement

1

Check | Try again | **Done**. Click any level to practice more. Completion is preserved.

✔ Expected:
(A) `getConnection`
(B) `prepareStatement`
(C) `setInt`
(D) `executeQuery`
(E) `RoomNumber`
(F) `close`
(G) `roomStatement`

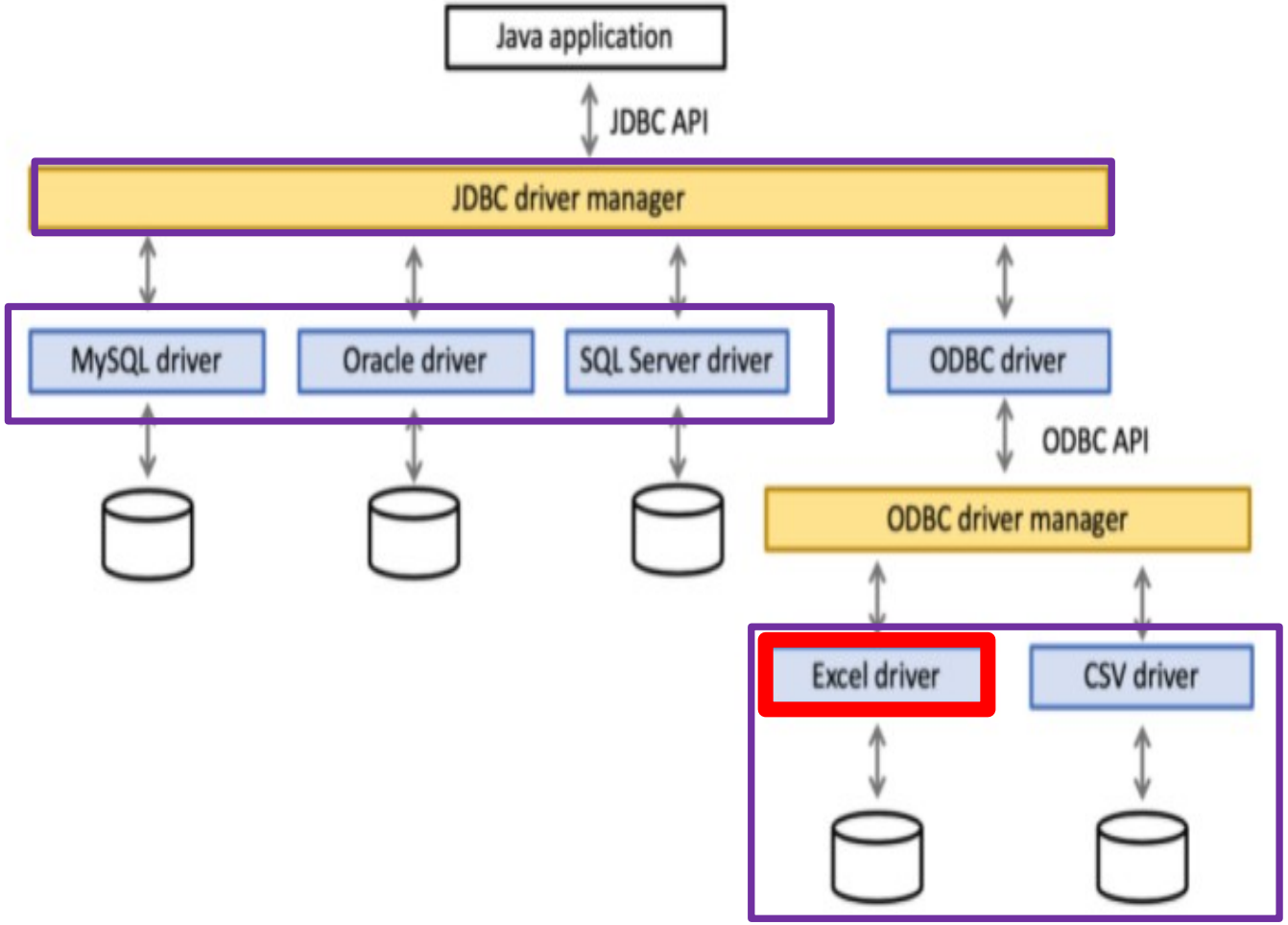
(A) `getConnection`: Establishes a connection, providing login and database information as parameters.
(B) `roomConn.prepareStatement(roomQuery)`: Creates a PreparedStatement object with the roomQuery.
(C) `roomStatement.setInt(1, 3)`: Assigns value 3, representing FloorNumber, to placeholder character ? in roomQuery.
(D) `roomStatement.executeQuery()`: Executes query and automatically converts Java data types to MySQL data types.
(E) `getInt("RoomNumber")`: Gets the results from the RoomNumber column.
(F) `resultSet.close()`: Releases the result set.
(G) `roomStatement.close()`: Releases the prepared statement resources.

View solution (Instructors only)

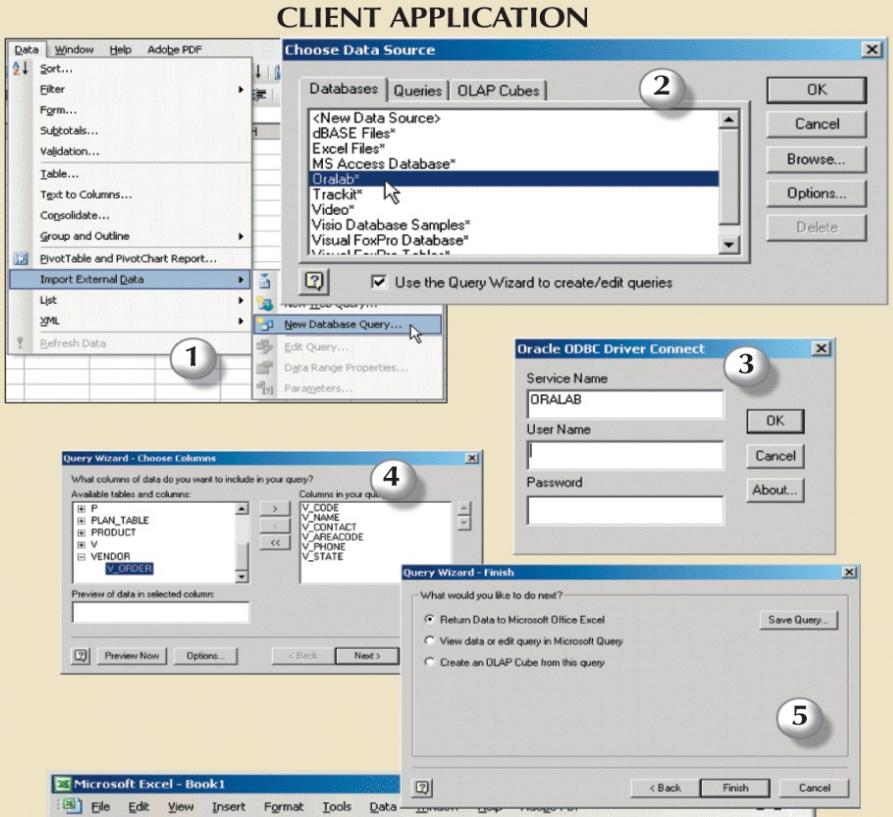
Yu, Jordan T

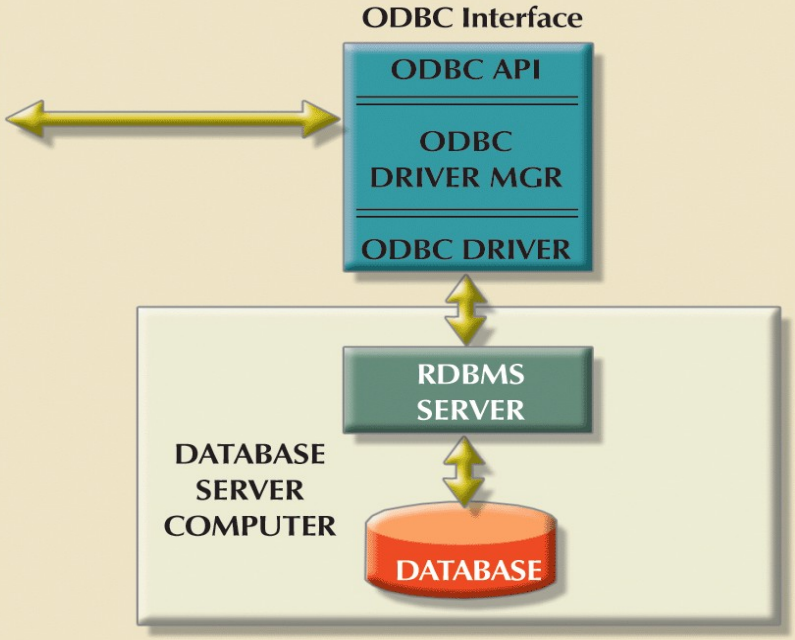
PARTICIPATION ACTIVITY

10.4.4: API architecture.



MS EXCEL uses ODBC to connect to an Oracle database





1. From Excel, select **Data**, **Import External Data** and **New Database Query** options to retrieve data from an Oracle RDBMS.
2. Select the **Oralab** ODBC data source (see Figure 14.3).
3. Enter the authentication parameters. ODBC uses the connection parameters to connect to the data source.
4. Select the table and the columns to use in the query.
5. Select **Return Data to Microsoft Office Excel**.
6. Excel uses the ODBC API to pass the SQL request down to the database. Oracle executes the request and generates a result set. Excel issues calls to the ODBC API to retreive the result set and populate the spreadsheet.

EXCEL & ODBC

10.5 Database programming with Python

Connections

This section describes database programming in Python and MySQL using Connector/Python. ***Connector/Python*** is a MySQL component based on the DB-API standard from the Python Software Foundation. This section assumes familiarity with the Python language.

Installing Connector/Python

The Connector/Python package must be installed on the computer running the Python program. Connector/Python can be downloaded from mysql.com and installed by running an installation program. Connector/Python can also be installed using pip from the Windows or Mac command line:

```
pip install mysql-connector-python
```

A Python program uses Connector/Python by importing the `mysql.connector` module. A ***module*** is a file containing Python classes, functions, or variables.

Python programs must connect to a database prior to executing queries. A connection is an object of the ***MySQLConnection*** class that is created with the ***mysql.connector.connect()*** function, specifying the database server address, database name, logon username, and password.

Creating a connection fails if the database is not found or logon credentials are invalid. Therefore, connections are usually created within the `try` block of a `try-except` statement. If the connection fails, the `except` block executes and typically prints an error message.

The Python program releases the connection when no longer needed with the ***connection.close()*** method.

PARTICIPATION ACTIVITY | 10.5.1: Creating a connection.

```
import mysql.connector
from mysql.connector import errorcode

try:
    reservationConnection = mysql.connector.connect(
        user='zamanzad',
        password='rjkai728',
        host='127.0.0.1',
        database='Reservation')

except mysql.connector.Error as err:
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
        print('Invalid credentials')
    elif err.errno == errorcode.ER_BAD_DB_ERROR:
        print('Database not found')
    else:
        print('Cannot connect to database:', err)

else:
    # Execute database operations...
    reservationConnection.close()
```

Python & MysqlConnection

```
import mysql.connector
from mysql.connector import errorcode

try:
    reservationConnection = mysql.connector.connect(
        user='ramanead',
        password='*jka1728',
        host='127.0.0.1',
        database='Reservation')

except mysql.connector.Error as err:
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
        print('Invalid credentials')
    elif err.errno == errorcode.ER_BAD_DB_ERROR:
        print('Database not found')
    else:
        print('Cannot connect to database:', err)

else:
    # Execute database operations...
    reservationConnection.close()
```

PARTICIPATION ACTIVITY | 10.5.2: Python connections. ?????

1) A connection must always be created within the try clause of a try-except-else statement.

   - ○ True
   - ◉ False

Correct

A connection can be created without a try-except-else statement. However, connections are usually created within a try clause so failures are processed in an except clause rather than causing the program to abort.

Concepts covered in this Lecture:

- Cursors
- Stored procedures

SQL **in application code**

- JDBC, ODBC (API)
- **SQL**J

- Embedded **SQL**
- Dynamic **SQL**

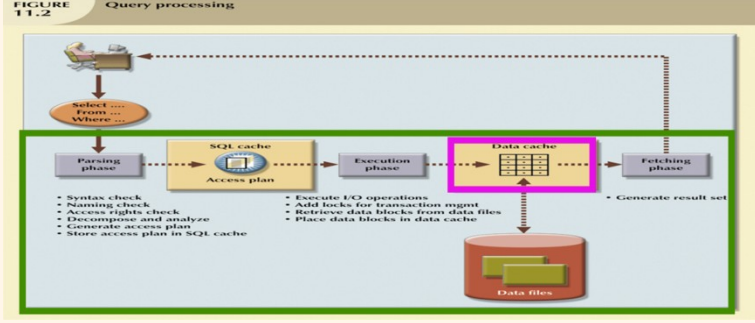
What is SQLJ – SQL Java?

- **SQLJ** is a set of **programming extensions** that allow a programmer using the **Java** programming language to embed statements that provide **SQL** database requests.

- **SQLJ** is similar to existing extensions for **SQL** that are provided for **C**, **FORTRAN**, and other programming languages.

- IBM, Oracle, and several other companies proposed **SQLJ** as a standard and as a simpler and easier-to-use alternative to JDBC.

SQLJ Code




```
int sid; String name; int rating;
// named iterator (Cursor) sailors
#sql iterator Sailors(int sid, String sname, int rating);
Sailors sailors;
// assume that the application sets rating
#sql sailors =
{
   SELECT sid, sname INTO :sid, :name FROM Sailors WHERE rating
   = :rating
};
// retrieve results
while (sailors.next())
{
   System.out.println(sailors.sid + " " + sailors.sname));
}
sailors.close();
```

Java

Concepts covered in this Lecture:

- Cursors
- Stored procedures

SQL in application code

- JDBC, ODBC (API)
- **SQL**J

- Embedded **SQL**
- Dynamic **SQL**

Embedded **SQL**

	Paradigm gap	Syntax gap	Object-oriented language	Applications	Compile steps	Network traffic	Database-independent
Embedded SQL	High	High	Limited	General	2	High	No

- Approach: Embed **SQL** in the host language.
  - A preprocessor converts the **SQL** statements into special **API** calls.
  - Then a regular compiler is used to compile the code.

- Language constructs:
  - Connecting to a database:
    EXEC SQL CONNECT
  - Declaring variables:
    EXEC SQL BEGIN (END) DECLARE SECTION
  - Statements:
    EXEC SQL Statement;

Embedding SQL in C: An Example

	Paradigm gap	Syntax gap	Object-oriented language	Applications	Compile steps	Network traffic	Database-independent
Embedded SQL	High	High	Limited	General	2	High	No

```
char SQLSTATE[6];
EXEC SQL BEGIN DECLARE SECTION
   char c_sname[20]; short c_minrating; float c_age;
EXEC SQL END DECLARE SECTION
c_minrating = random();
EXEC SQL DECLARE sinfo CURSOR FOR
   SELECT S.sname, S.age
    FROM Sailors S
   WHERE S.rating > :c_minrating
   ORDER BY S.sname;
do {
   EXEC SQL FETCH sinfo INTO :c_sname, :c_age;
   printf("%s is %d years old\n", c_sname, c_age);
} while (SQLSTATE != '02000');
EXEC SQL CLOSE sinfo;
```

10.3 Embedded SQL

	Paradigm gap	Syntax gap	Object-oriented language	Applications	Compile steps	Network traffic	Database-independent
Embedded SQL	High	High	Limited	General	2	High	No

PARTICIPATION ACTIVITY 10.3.1: Embedded SQL in a C program.

```
int main() {

   /* Other C statements */

   printf("Updating flight number 308\n");

      EXEC AirlineName = 'British Airways', AirportCode = 'JFK'
      WHERE FlightNumber = 308;

   /* Other C statements */

   return 0;
}
```

precompiler generates C statements and function calls

C

Concepts covered in this Lecture:

- Cursors
- Stored procedures

SQL in application code

- JDBC, ODBC (API)
- **SQL**J

- Embedded **SQL**
- Dynamic **SQL**

*Dynamic **SQL***

- **SQL** query strings **are not always known at compile time** (e.g., spreadsheet, graphical **DBMS** front-end)

- Allow construction of **SQL** statements on-the-fly

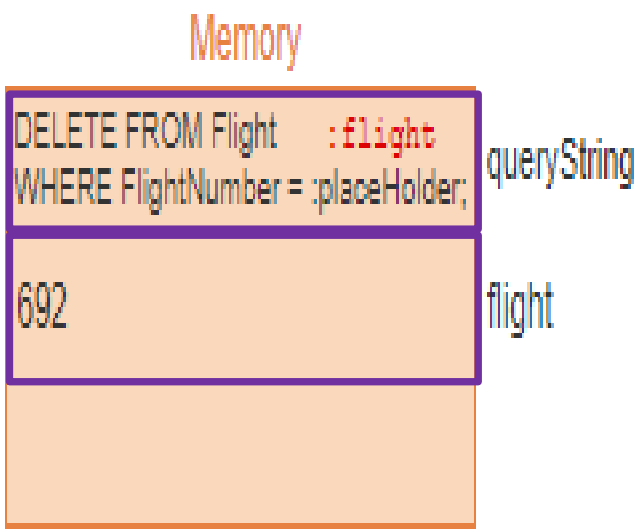
- Example:

```
char c_sqlstring[ ]={“DELETE FROM Sailors WHERE rating>5”};
EXEC SQL PREPARE readytogo FROM :c_sqlstring;
EXEC SQL EXECUTE readytogo;
```

PARTICIPATION ACTIVITY

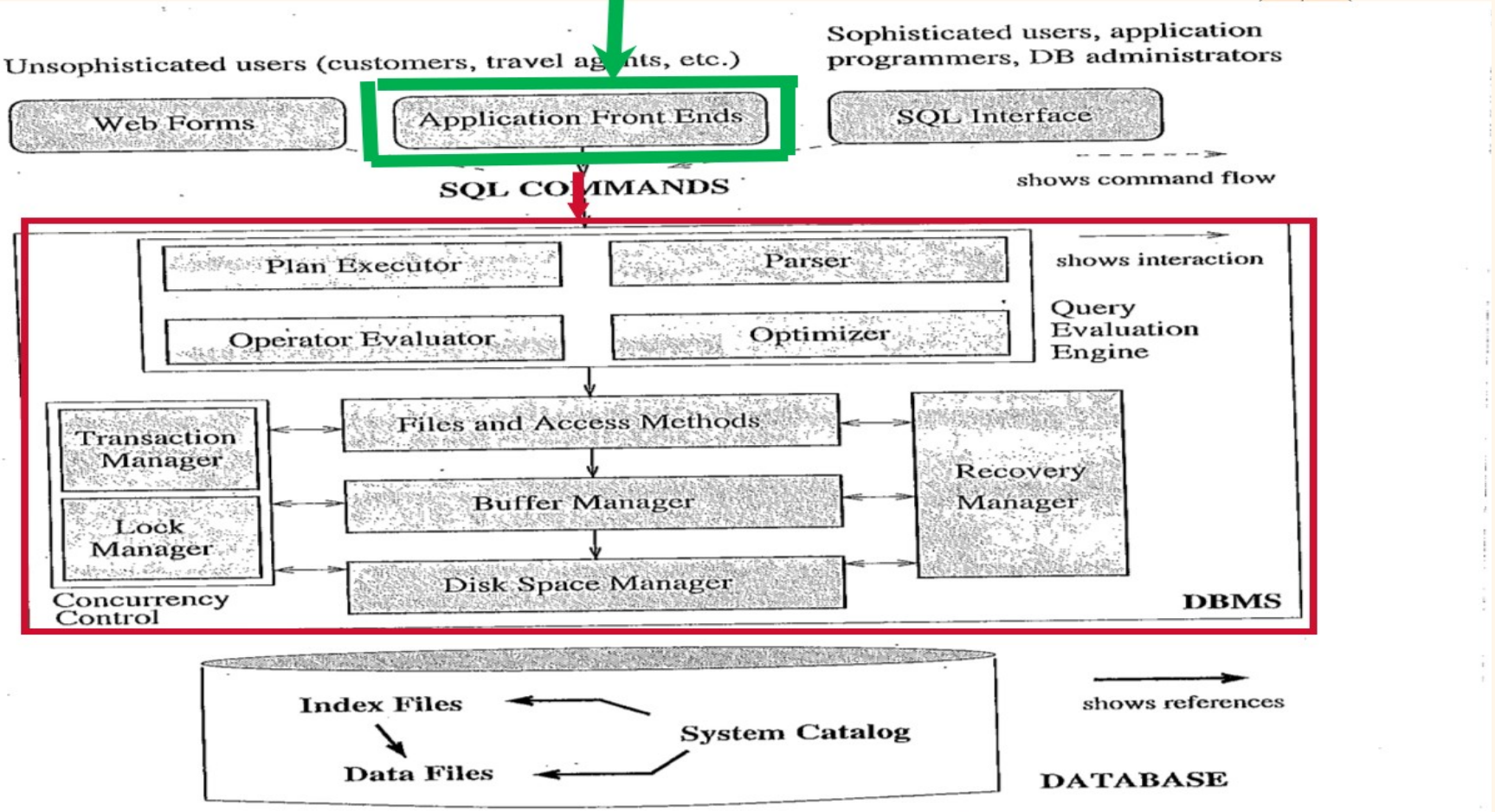
10.3.9: Dynamic SQL in a C program fragment.

```
EXEC SQL BEGIN DECLARE SECTION;
   char queryString[200];
   int flight;
EXEC SQL END DECLARE SECTION;

printf("Enter query: ");
scanf("%s", queryString);
EXEC SQL PREPARE QueryName FROM :queryString;

printf("Enter flight number: ");
scanf("%d", &flight);
EXEC SQL EXECUTE QueryName USING :flight;
```



WOW! Application Programming Technologies

DBMS **APPLICATION SOFTWARE**



APPLICATIONS Practice Questions

Question	Which of the following is **not** one of the ways users can be provided with dynamic access to an Oracle database?
Answer	PL/SQL Web Toolkit ?????
	Tiered Architectures
	PL/SQL Server Pages
	JDBC

2

Question	Which of the following is used to access the database server at time of executing the program and get the data from the server accordingly?
Answer	Embedded SQL
	Dynamic SQL ?????
	SQL declarations
	SQL data analysis

Dynamic SQL

- SQL query strings **are not always known at compile time** (e.g., spreadsheet, graphical **DBMS** front-end)
- Allow construction of SQL statements on-the-fly
- Example:

```
char c_sqlstring[ ]={"DELETE FROM Sailors WHERE rating>5"};
EXEC SQL PREPARE readytogo FROM :c_sqlstring;
EXEC SQL EXECUTE readytogo;
```

Question	Which of the following header must be included in java program to establish database connectivity using JDBC ?
Answer	import java.sql.*; ?????
	import java.sql.odbc.jdbc.*;
	import java.jdbc.*;
	import java.sql.jdbc.*;

Question	DriverManager.getConnection(_____, _____, _____)
	What are the parameters that are included ?
Answer	URL or machine name where server runs, Password, User ID ?????
	URL or machine name where server runs, User ID, Password
	User ID, Password, URL or machine name where server runs
	Password, URL or machine name where server runs, User ID

What is missing from the join statement?

```
SELECT EmployeeName, Salary FROM Employee, Department WHERE Salary > 50000;
```

Left and right tables are not specified.

A column from Employee is not compared to a column from Departement

No columns from Department appear in the SELECT clause.

?????

10. SET 2 - 3:APPLICATIONS

The EXEC SQL keyword is used in procedural SQL.

True

False

Embedded SQL

- Approach: Embed SQL in the host language.
  - A preprocessor converts the SQL statements into special API calls.
  - Then a regular compiler is used to compile the code.

- Language constructs:
  - Connecting to a database:
    EXEC SQL CONNECT
  - Declaring variables:
    EXEC SQL BEGIN (END) DECLARE SECTION
  - Statements:
    EXEC SQL Statement;

?????

Refer to the Country table below.

Country

Code	Name	HeadOfState	IndepYear	Population
ABW	Aruba	Beatrix	NULL	103000
AIA	Anguilla	Elizabeth II	1920	NULL
AFG	Afghanistan	Mohammad Omar	1919	22720000
AGO	Angola	Jose dos Santos	1975	12878000

How many rows are returned?

```
SELECT *  FROM Country  WHERE Population IS NULL;
```

0

1

3

?????

The _____ statement moves a cursor to the next row of a result table.

FETCH

Cursors

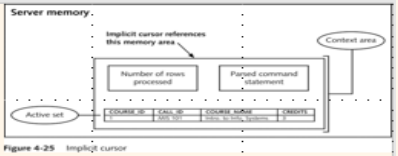


Figure 4-25 Implicit cursor

- Can declare a **cursor** on a **Relation** or **SQL query statement** (which generates a **Relation**).
- Can *open* a **cursor**, and repeatedly ***fetch*** a **tuple**, then *move* the **cursor**, until all **tuples** have been retrieved.
  - Can use a special clause, called ORDER BY, in queries that are accessed through a **cursor**, to control the order in which **tuples** are returned.
    - Fields in ORDER BY clause **must also appear** in SELECT clause.
  - The ORDER BY clause, which orders answer **tuples**, is *only* allowed in the context of a **cursor**.
- Can also **modify/delete tuple** pointed to by a **cursor**.

13

?????

Which API is often used with the C# language ?

ADO.NET

JDBC

DB-API

Table 10.4.1: Leading database APIs.

Acronym	Derivation	Original Sponsor	Initial release	Application language	MySQL Connector
ODBC	Open Database Connectivity	Microsoft	1992	Numerous	Connector/ODBC
JDBC	Java Database Connectivity	Sun Microsystems (now Oracle)	1997	Java	Connector/J
DB-API	Database API	Python Software Foundation	1996	Python	Connector/Python
ADO.NET	ActiveX Data Objects	Microsoft	2002	.NET languages	Connector/NET
PDO	PHP Data Objects	The PHP Group	2005	PHP	none

?????

Refer to the City table.

City

ID	Name	CountryCode	District	Population
207	Rio de Janeiro	BRA	Rio de Janeiro	5598953
462	Manchester	GBR	England	430000
554	Santiago de Chile	CHL	Santiago	4703954
654	Barcelona	ESP	Katalonia	1503451
826	Valencia	PHL	Northern Mindanao	147924
1025	Delhi	IND	Delhi	7206704

What cities are selected?

```
SELECT *   FROM city   WHERE ID < 300      OR Population > 7500000;
```

All cities

Delhi and Rio de Janeiro

Rio de Janeiro

?????

The following stored procedure awards bonuses based on employee performance ratings:

```
CREATE PROCEDURE AwardBonus()
BEGIN

   DECLARE bonusAmount INT;
   DECLARE ratingCount INT;
   ___A___ rating INT DEFAULT 10;

   ___B___ rating > 0 DO

   SELECT COUNT(*)
   ___C___ ratingCount
   FROM Employee
   WHERE PerformanceRating = rating;

   ___D___ ratingCount > 80
   SET bonusAmount = 1000 * rating;
   ELSE
   SET bonusAmount = 100 * rating;
   END IF;

   UPDATE Employee
   SET EmployeeBonus = bonusAmount
   WHERE PerformanceRating = rating;

   ___E___ rating = rating - 1;

   END WHILE;

END;
```

What is **keyword C**?

INTO

Procedural SQL

```
CREATE PROCEDURE FlightCount(IN airline VARCHAR(20), OUT quantity INT)
   SELECT COUNT(*)
   INTO quantity
   FROM Flight
   WHERE AirlineName = airline;
```

?????

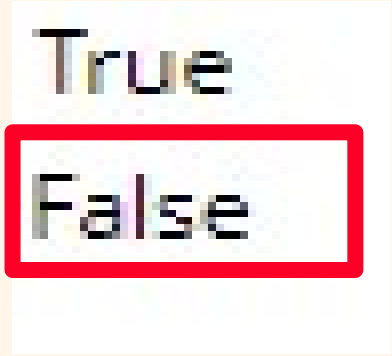
3.1415 can be stored as an INT type in Oracle Database.

True

False

?????

A stored procedure is compiled every time the procedure is called.



?????

Which statement is correct embedded SQL syntax?

EXEC SQL INSERT INTO Employee (ID, Name, Salary) VALUES (2538, 'Lisa Ellison', 45000);

EXEC SQL INSERT INTO Employee (ID, Name, Salary);
EXEC SQL VALUES (2538, 'Lisa Ellison', 45000);

EXEC SQL INSERT INTO Employee (ID, Name, Salary) VALUES (2538, 'Lisa Ellison', 45000)

INSERT INTO Employee (ID, Name, Salary) VALUES (2538, 'Lisa Ellison', 45000);

Embedded SQL

- Approach: Embed SQL in the host language.
  - A preprocessor converts the SQL statements into special API calls.
  - Then a regular compiler is used to compile the code.
- Language constructs:
  - Connecting to a database:
    EXEC SQL CONNECT
  - Declaring variables:
    EXEC SQL BEGIN (END) DECLARE SECTION
  - Statements:
    EXEC SQL Statement;

?????

Refer to the following CREATE VIEW statement and view table.

Faculty

ID	Name	DeptCode
1	Grayson	ART
2	Wayne	ART
3	Stark	COMP
4	Parker	MATH
5	Banner	MATH
6	Quinn	MATH
7	Grey	NULL

Department

Code	Name	ChairID
ART	Art Department	2
COMP	Computer Science Department	3
ENG	English Department	NULL
HIST	History Department	NULL
MATH	Math Department	6

```
CREATE VIEW DepartmentView  AS SELECT Department.Name AS DepartmentName, Faculty.Name AS Chair     FROM Department, Faculty     WHERE ChairID = Faculty.ID;
```

DepartmentView

DepartmentName	Chair
Art Department	Wayne
Computer Science Department	Stark
Math Department	Quinn

ID in Faculty is the PRIMARY KEY. Code in Department is the PRIMARY KEY.

If DepartmentView is a materialized view, against what tables is the query executed?

```
SELECT * FROM DepartmentView;
```

Department only

DepartmentView

Faculty and Department

?????

Which line assigns the correct tuple for the UPDATE statement?

```
flightQuery = ('UPDATE Flight ' 'SET AirlineName = %s ' 'WHERE FlightNumber = %s')
flightCursor.execute(flightQuery, flightData)
```

```
flightData = ('United Airlines', 105)
```

```
flightData = ('United Airlines', 'PEK')
```

```
flightData = ('United Airlines', 'PEK', 105)
```

?????

A precompiler translates embedded SQL to:

SQL

Host language

Machine language

Embedded SQL

- Approach: Embed SQL in the host language.
  - A preprocessor converts the SQL statements into special API calls.
  - Then a regular compiler is used to compile the code.

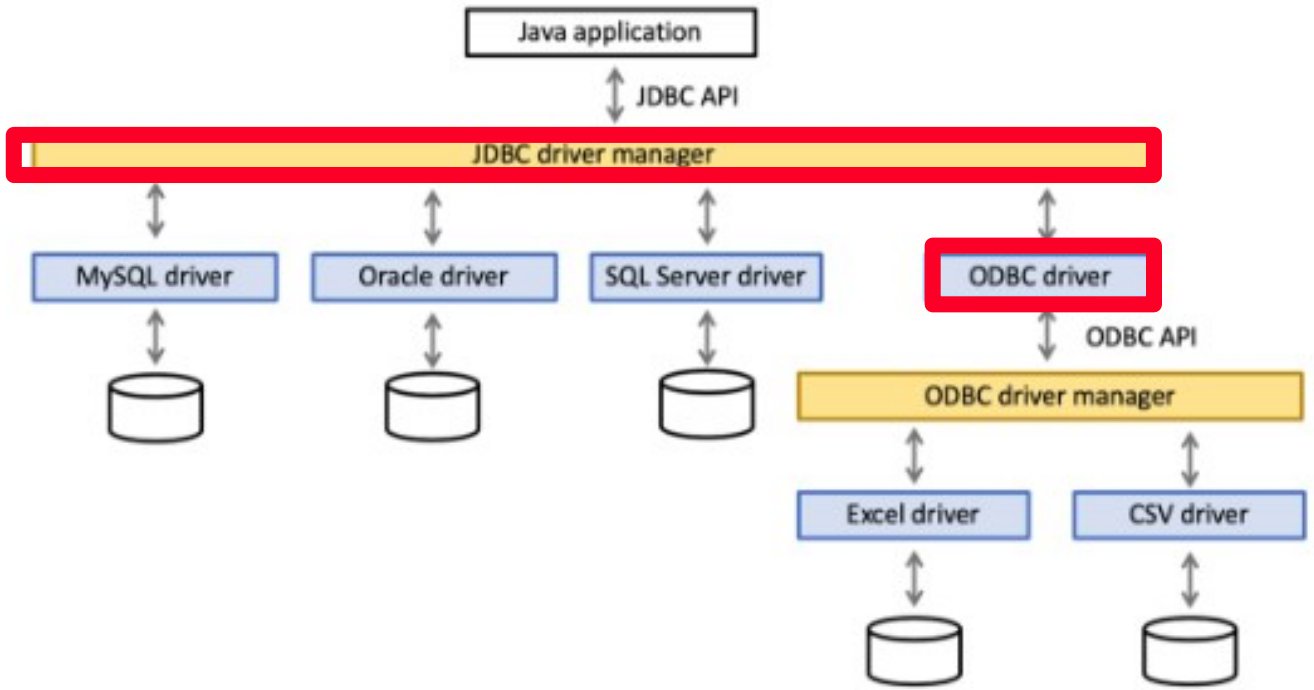
- Language constructs:
  - Connecting to a database:
    EXEC SQL CONNECT
  - Declaring variables:
    EXEC SQL BEGIN (END) DECLARE SECTION
  - Statements:
    EXEC SQL Statement;

?????

In the animation below, the secondary API is ODBC.

PARTICIPATION ACTIVITY | 10.4.4: API architecture.

Start 2x speed



True

False

?????

Choose the correct **SELECT** statement that returns the given results from the Auto table below.

Auto

ID	Make	Model	Type	Year	Price
1	Toyota	Camry	sedan	2015	9800
2	Ford	Escape	crossover	2015	15900
3	Honda	Civic	sedan	2016	10200
4	Volkswagen	Golf	compact	2014	8800
5	Toyota	RAV4	crossover	2016	12800
6	Toyota	4Runner	suv	2015	16900
7	Honda	CR-V	crossover	2016	17900

Result obtained: 2

```
SELECT COUNT(*) FROM Auto WHERE Price < 10000;

SELECT COUNT(*) FROM Auto WHERE Price > 10000;

SELECT COUNT(*) FROM Auto;
```

?????

PARTICIPATION ACTIVITY | 8.8.1: Adding primary key constraints to tables.

```
CREATE TABLE Employee (
    ID          SMALLINT UNSIGNED,
    Name        VARCHAR(60),
    Salary      DECIMAL(7,2),
    PRIMARY KEY (ID)
);
```

Employee

ID	Name	Salary
2538	Lisa Ellison	45000
5384	Sam Snead	30400
6381	Maria Rodriguez	92300

```
CREATE TABLE Family (
    ID              SMALLINT UNSIGNED,
    Number          SMALLINT UNSIGNED,
    Relationship    VARCHAR(20),
    Name            VARCHAR(60),
    PRIMARY KEY(ID, Number)
);
```

Family

ID	Number	Relationship	Name
2538	1	Spouse	Henry Ellison
2538	2	Son	Edward Ellison
6381	1	Spouse	Jose Rodriguez
6381	2	Daughter	Gina Rodriguez
6381	3	Daughter	Clara Rodriguez

Only one column may be listed in a PRIMARY KEY constraint.

True

False

?????

Choose the correct ORDER BY clause to produce the results in each question.

```
SELECT Name, District, Population
FROM City
_______;
```

City

ID	Name	CountryCode	District	Population
301	Embu	BRA	São Paulo	222223
302	Mossoró	BRA	Rio Grande do Norte	214901
303	Várzea Grande	BRA	Mato Grosso	214435
304	Petrolina	BRA	Pernambuco	210540
305	Barueri	BRA	São Paulo	208426

```
Embu             São Paulo            222223
Barueri          São Paulo            208426
Mossoró          Rio Grande do Norte  214901
Petrolina        Pernambuco           210540
Várzea Grande    Mato Grosso          214435
```

ORDER BY District

ORDER BY District DESC

ORDER BY Name DESC

?????

Refer to the following CREATE VIEW statement and view table.

Faculty

ID	FacultyName	Code
1	Grayson	ART
2	Wayne	ART
3	Stark	COMP
4	Parker	MATH
5	Banner	MATH
6	Quinn	MATH
7	Grey	NULL

Department

Code	DepartmentName
ART	Art Department
COMP	Computer Science Department
ENG	English Department
HIST	History Department
MATH	Math Department

```
CREATE VIEW __A__  AS SELECT FacultyName AS __B__, __C__ AS Assignment  FROM Faculty, Department  WHERE Faculty.__D__ = Department.Code  AND Department.Code = '__E__';
```

MathFacultyView

Professor	Assignment
Parker	Math Department
Banner	Math Department
Quinn	Math Department

What is value E?

MATH

?????

10. SET 2 - 3:APPLICATIONS

The PHP code below throws an exception at line _____.

```
1. $sql = "INSERT INTO Teacher " .
2.        "(LastName, FirstName, Department) " .
3.         "VALUES (? ,?, ?)";
4.
5. $stmt = $pdo->prepare($sql);
6. $stmt->bindValue(1, $_POST["lastName"]);
7. $stmt->bindValue(2, $_POST["firstName"]);
8. $stmt->execute();
9.
10. echo "Teacher is added.";
```

a. 1

b. 5

c. 8

d. 10

?????

A/An _____ will retrieve rows from a query that has multiple rows in the result set.

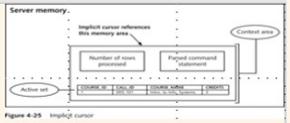
a. cursor

b. shared variable

c. index

d. fetch

Cursors



- Can declare a **cursor** on a **Relation** or **SQL query statement** (which generates a **Relation**).
- Can *open* a **cursor**, and repeatedly ***fetch*** a **tuple**, then *move* the **cursor**, until all **tuples** have been retrieved.
  - Can use a special clause, called ORDER BY, in queries that are accessed through a **cursor**, to control the order in which **tuples** are returned.
    - Fields in ORDER BY clause **must also appear** in SELECT clause.
  - The ORDER BY clause, which orders answer **tuples**, is *only* allowed in the context of a **cursor**.
- Can also **modify/delete tuple** pointed to by a **cursor**.

13

?????

What is the correct statement for creating a database called reservationDB?

a. DATABASE CREATE reservationDB;

b. CREATE DB reservationDB;

c. CREATE reservationDB DATABASE;

d. CREATE DATABASE reservationDB;

?????

Which SQL statement deletes the table Supplier?

a. DROP Supplier;

b. DROP TABLE Supplier;

c. DELETE TABLE Supplier;

d. DROP Supplier TABLE;

?????

When a user interacts with a database, they can use a ____ to modify data with commands.

a. query processor

b. query language

c. data dictionary

d. transaction manager

?????

When using the fetch() in a SELECT statement, each call to fetch() returns the next row from the result table. If it reaches the last row, the next returned value is ____.

a. NULL

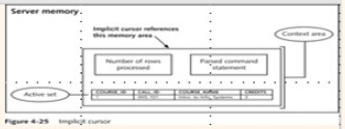
b. false

c. LAST_ROW_VALUE

d. true

?????

Cursors



- Can declare a **cursor** on a **Relation** or **SQL query statement** (which generates a **Relation**).
- Can *open* a **cursor**, and repeatedly ***fetch*** a **tuple**, then *move* the **cursor**, until all **tuples** have been retrieved.
  - Can use a special clause, called ORDER BY, in queries that are accessed through a **cursor**, to control the order in which **tuples** are returned.
    - Fields in ORDER BY clause **must also appear** in SELECT clause.
  - The ORDER BY clause, which orders answer **tuples**, is *only* allowed in the context of a **cursor**.
- Can also **modify/delete tuple** pointed to by a **cursor**.

13

At 5:00 PM .

02.21.2024 (11 - We)	ZyBook SET 2 - 3	Set 2 LECTURE 10 APPLICATIONS

VH work on
SET 2 – 3: APPLICATIONS

Next

VH, unHIDE

From 5:05 to 5:15 PM – 5 minutes.

02.21.2024 (11 - We)	ZyBook SET 2- 3	Set 2 LECTURE 10 APPLICATIONS

CLASS PARTICIPATION 20 points — 20% of Total

APPLICATIONS

Class 11 END PARTICIPATION

Not available until Feb 21 at 5:05pm | Due Feb 21 at 5:15pm | 100 pts

VH, publish

This is a synchronous online class.

Attendance is required.

Recording or distribution of class materials is prohibited.

1. At the beginning of selected classes there is an **assessment in the first 10 minutes**. (**beige** BOX in the Detailed Syllabus)
2. At the end of selected classes there is an **assessment in the last 10 minutes**. (**blue** BOX in the Detailed Syllabus)
3. ZyBook sections will be downloaded and used for 30% of Total Score on the dates specified in the Detailed Syllabus.
4. EXAMS are in CANVAS. **No late EXAMS**.
5. I have to be **present in TEAMS in order to take any graded assignment assigned during that class.**

At 5:15 PM.

End Class 11

VH, unhide ZyBook Section 11.

VH, Download Attendance Report
Rename it:
2.21.2024 Attendance Report FINAL

VH, upload Class 11 to CANVAS.