Lecture 8 Access SQL from the Host Language

CS211 - Introduction to Database

Motivation

```
IF condition THEN
  SQL-Query 1
ELSE
  SQL-Query 2
END IF
```

Result = SQL-Query

Conduct complex computation on the result

Interaction with users

Two approaches

- Dynamic SQL
  - A general-purpose program can connect to and communicate with a database server using a collection of functions or methods.
  - Allows the host to construct an SQL query as a string at runtime, submit the query, and then retrieve the result
  - C/C++, Java, Python, …

- Embedded SQL
  - The SQL statements are identified at compile time using a pre-compiler
  - C/C++, Java, COBOL, …

An example of Embedded SQL

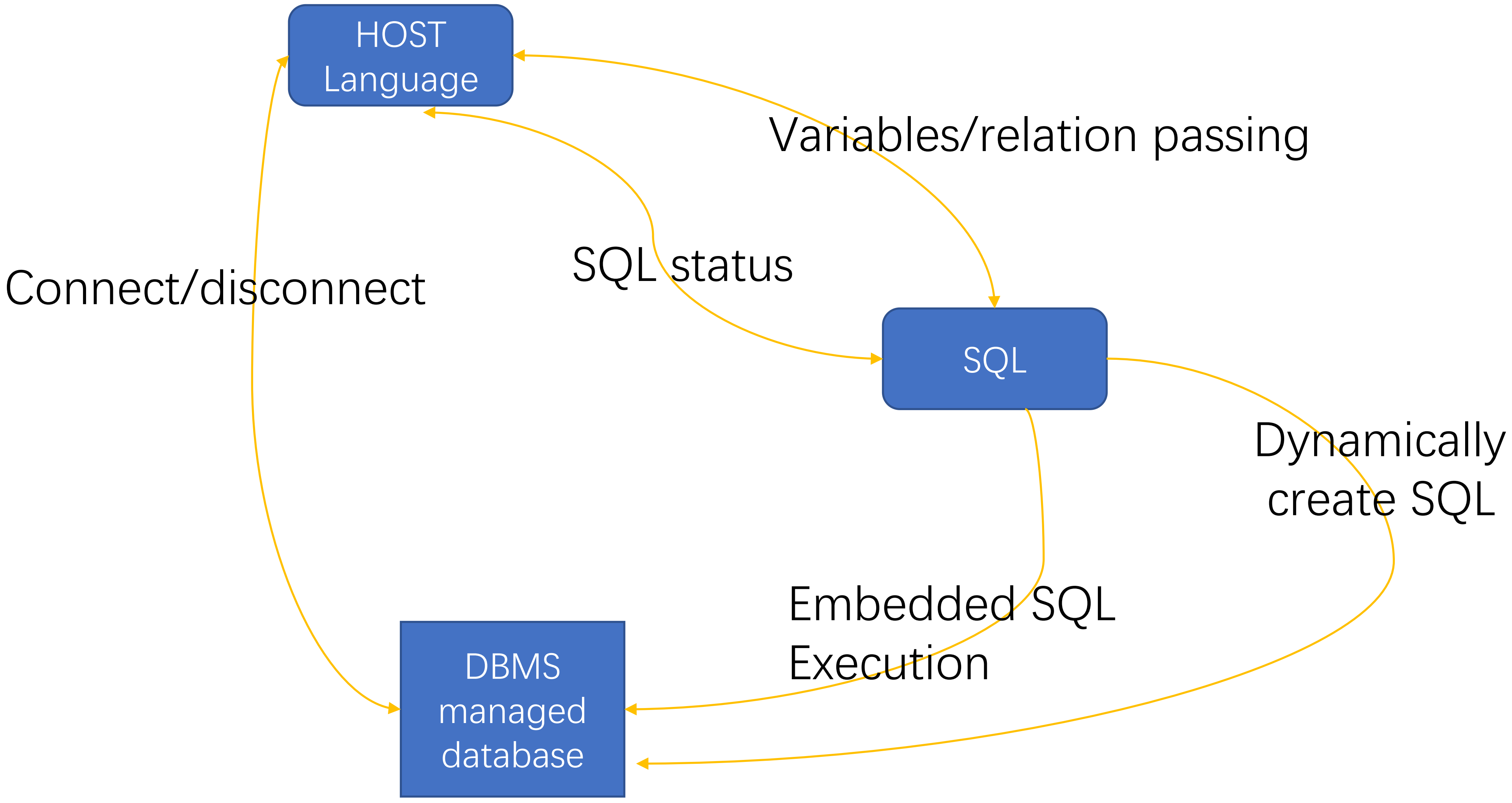
SQL: select sName, age from student where sName = 'John'

- Host: C

```
exec sql select sName, age into :vName, :vAge
              from student where sName = 'John'
```

exec sql: let C compiler know that it should be handled with the database pre-compiler

into: :variable-names


HOST Language
Variables/relation passing
SQL status
Connect/disconnect
SQL
Dynamically create SQL
Embedded SQL Execution
DBMS managed database

Embedded SQL: Database connect

SQL standard

```
exec sql connect to target-server as connect-name user user-name;
```

```
exec sql connect to default;
```

Oracle

```
exec sql connect :user_name identified by :user_pwd;
```

DB2

```
exec sql connect to mydb user :user_name using :user_pwd;
```

Embedded SQL: Database disconnect

SQL standard

```
exec sql disconnect connect-name;
```

```
exec sql connect current;
```

Oracle

```
exec sql commit release;  / exec sql rollback release;
```

DB2

```
exec sql connect reset;  / exec sql disconnect current;
```

Embedded SQL: commit & rollback

confirm

```
exec sql commit work;
```

```
exec sql rollback work;
```

Why is the confirmation required?

Transaction

- a collection of operations that form a single logical unit of work
  - One or more SQL statements

```
Begin Transaction
  exec sql …
  …
  exec sql …
  exec sql commit work | rollback work
End Transaction
```

Transaction - example

- Bank - transaction T
  - transfer 1,000 from account A to account B

```
Read A;              # read balance from database to buffer
A = A – 1000;
Write A;
Read B;
B = B + 1000;
Write B;
```

Dynamic SQL – python example

- **Python + mariaDB + mariadb package**

Install package for python	`pip3 install mariadb;`
Import the package	`import mariadb`

Dynamic SQL – python example

Connect Python program to database

```
conn=mariadb.connect( host='127.0.0.1’,
                      port=3306,
                      user=‘uid’,
                      password=‘pwd’,
                      database=‘exampledb')
conn.close()
```

Dynamic SQL – python example

Python accesses the database through a Cursor:

Get and Execute SQL

```
cur = conn.cursor()

cur.execute("SELECT sName FROM student")
```

Download mariadb-python.ipynb from course moodle page