雨课堂
Rain Classroom

本次课程是

线上+线下
融合式教学

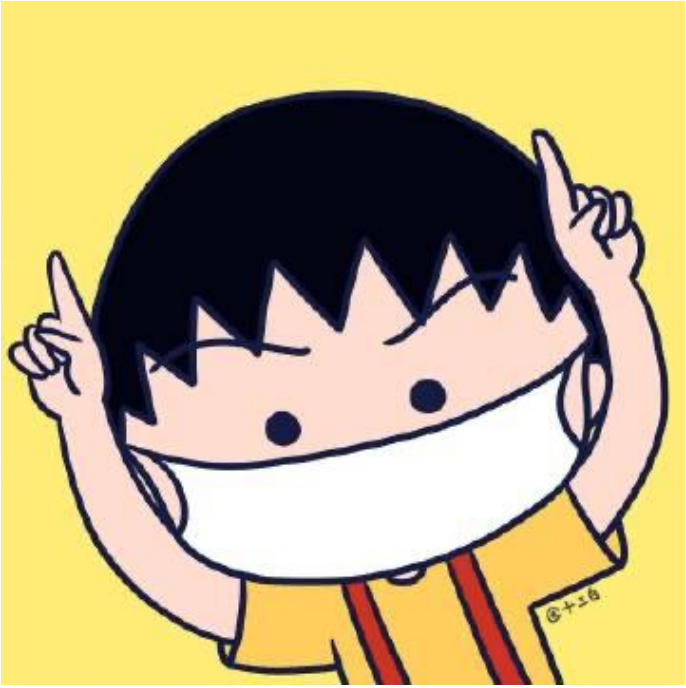
请**现场**的同学们：

1. 打开雨课堂，点击页面右下角喇叭按钮调至静音状态

请**远程上课**的同学们：

1. 打开雨课堂，点击页面右下角喇叭按钮调至静音状态
2. 打开“瞩目”（会议室：182 943 865；密码：见学堂公告），进入会议室，并关闭麦克风

请在教室内佩戴口罩



Database Concepts (III)

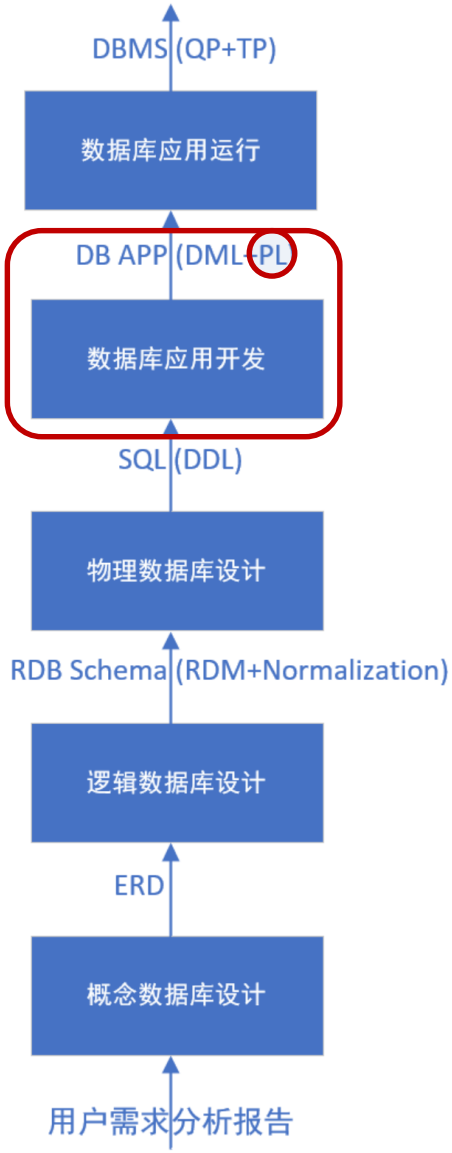
Structured Query Language

Chaokun Wang

School of Software, Tsinghua University

chaokun@tsinghua.edu.cn

April 25, 2022



Outline

- Introduction to SQL
- Data Manipulation Language*
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
- Data Definition Language*
  - Data Types
  - Schema
  - Table
  - Index
  - View
  - Transaction
- Procedural SQL

Stored Procedures

- Named collection of procedural and SQL statements

存储过程：就是一系列SQL语句的集合体，我们可以理解为一个封装单元，这个单元可以有出入参数，也可以没有

  – Stored in the database
  – Can be used to encapsulate and represent business transactions
- Advantages
  – Reduce network traffic and increase performance
  – Decrease code duplication by means of code isolation and code sharing

https://blog.csdn.net/u013408431/article/details/73275935?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216513859501678168396163 0%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=165138595016781683961630&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-2-73275935.142^v9^pc_search_result_cache,157^v4^control&utm_term=%E5%AD%98%E5%82%A8%E8%BF%87%E7%A8%8B%E5%92%8C%E5%87%BD%E6%95%B0%E7%9A%84%E5%8C%BA%E5%88%AB&spm=1018.2226.3001.4187

这是关于存储过程的一个CSDN的教学

Stored Procedures: Sample (In PG)

存储过程名称后面必须加括号，就算没有参数传递也要加

```
CREATE OR REPLACE PROCEDURE addStaff(inputNum int) AS $$
DECLARE myNum smallint = 0;
BEGIN
WHILE myNum < inputNum LOOP
INSERT INTO Staff VALUES ('SG16'||to_char(myNum,'9999'), 'Alan',
    'Brown', 'Assistant', 'M', DATE '1957-05-25', 8300+myNum,'B003');
myNum := myNum + 1;
END LOOP;
END
$$ LANGUAGE plpgsql;
```

$$用来包裹住函数(存储过程)的主体语句

这一步是在声明变量

存储过程里面可以用区块、条件以及循环

这个case的意义在于……？

Query Editor

```
call addStaff(10);
```

Query Editor

```
select * from staff;
```

Data Output | Explain | Messages | Notifications | Query History

	staffno [PK] character varying (255)	fname character varying (255)	lname character varying (255)	position character varying (255)	sex character varying (255)	dob character varying (255)	salary numeric (7,2)	branchno character vary
1	SG16 0	Alan	Brown	Assistant	M	1957-05-25	8300.00	B003
2	SG16 1	Alan	Brown	Assistant	M	1957-05-25	8301.00	B003
3	SG16 2	Alan	Brown	Assistant	M	1957-05-25	8302.00	B003
4	SG16 3	Alan	Brown	Assistant	M	1957-05-25	8303.00	B003
5	SG16 4	Alan	Brown	Assistant	M	1957-05-25	8304.00	B003
6	SG16 5	Alan	Brown	Assistant	M	1957-05-25	8305.00	B003
7	SG16 6	Alan	Brown	Assistant	M	1957-05-25	8306.00	B003
8	SG16 7	Alan	Brown	Assistant	M	1957-05-25	8307.00	B003
9	SG16 8	Alan	Brown	Assistant	M	1957-05-25	8308.00	B003

PL/SQL Stored Functions

- Stored function: named group of procedural and SQL statements that returns a value
  - Indicated by a RETURN statement in its program code 函数是可以用在SELECT里面的（因为有返回值）
- Can be invoked only from within stored procedures or triggers
  - Cannot be invoked from SQL statements unless the function follows some very specific compliance rules

PL/pgSQL Stored Functions: Sample (In PG)

Query Editor

function是有返回值的，和存储过程比起来的话针对性更强一点

```
CREATE OR REPLACE FUNCTION sales_tax(subtotal real) RETURNS real AS $$
BEGIN
    IF (subtotal < 1000) THEN
        RETURN subtotal * 0.06;
    ELSE
        RETURN subtotal * 0.10;
    END IF;
END
$$ LANGUAGE plpgsql;
```

主体部分必须用$$BEGIN...END$$宝珠

这里意思是用的语言

Data Output | Explain | Messages | Notifications | Query History

```
CREATE FUNCTION
```

写了一个函数，他的结果可以直接被SELECT

Query Editor

```
select sales_tax(500);
```

Data Output | Explain | Messages | Noti

	sales_tax real
1	30

Query Editor

```
select sales_tax(5000);
```

Data Output | Explain | Messages | N

	sales_tax real
1	500

PL/SQL Processing with Cursors (游标)

一般我们的返回值都是一个含有多个记录的集合，游标机制允许用户逐行的访问这些记录并且按照自己的意愿来显示和处理这些记录

- Cursor: special construct used to hold data rows returned by an SQL query
  - Implicit cursor: automatically created when SQL statement returns only one value 隐性游标
  - 明确的 Explicit cursor: holds the output of an SQL statement that may return two or more rows
  - Syntax:

    CURSOR *cursor_name* IS *select-query*;(Oracle)

    cursor_name CURSOR FOR *select-query*; (PG)
- Cursor-style processing involves retrieving data from the cursor one row at a time
  - Current row is copied to PL/SQL variables

PL/pgSQL Processing with Cursors: Sample (In PG)

· 步骤：
声明游标-打开游标-使用游标操作数据-关闭游标

Query Editor

我们这里写的是一个过程

```
1  CREATE OR REPLACE PROCEDURE PRC_CURSOR_EXAMPLE() AS $$
2  DECLARE
3     v_staffno STAFF.STAFFNO%TYPE;
4     v_salary STAFF.SALARY%TYPE;
5     PROD_CURSOR CURSOR FOR
6        SELECT STAFFNO, SALARY
7        FROM STAFF
8        WHERE FNAME = 'Alan';
9
10 BEGIN
11 RAISE INFO '=====================================';
12 OPEN PROD_CURSOR;
13 LOOP
14    FETCH PROD_CURSOR INTO v_staffno, v_salary;
15    EXIT WHEN NOT FOUND;
16    RAISE INFO '% -> %', v_staffno, v_salary;
17 END LOOP;
18 RAISE INFO '=====================================';
19 RAISE INFO '--- END OF REPORT ----';
20 CLOSE PROD_CURSOR;
21 END;
22 $$ LANGUAGE plpgsql;
23
```

声明过程，这里我们声明了两个变量&一个游标

%TYPE的意思是，我的v_staffno的数据类型了STAFF.STAFFNO是一样的

游标名

FETCH syntax:
FETCH cursor_name INTO variable list;

使用游标之前有一个OPEN的动作

FETCH是获取游标当前指针的记录，并且用INTO传递给指定列表

RAISE是用来打印字符串的函数，其中%为参数占位符，按顺序与后面的参数对应

RAISE相关后面的[level]制定了错误的严重性，允许的级别有debug/ log/ info/ notice/ warning和exception，默认级别是exception（会终止当前事务）

Data Output | Explain | Messages | Notifications | Query History

```
CREATE PROCEDURE
```

Query Editor

```
1  CALL PRC_CURSOR_EXAMPLE();
```

Data Output | Explain | Messages | Notifications | Que

```
信息:  =====================================
信息:  SG16    0 -> 8300.00
信息:  SG16    1 -> 8301.00
信息:  SG16    2 -> 8302.00
信息:  SG16    3 -> 8303.00
信息:  SG16    4 -> 8304.00
信息:  SG16    5 -> 8305.00
信息:  SG16    6 -> 8306.00
信息:  SG16    7 -> 8307.00
信息:  SG16    8 -> 8308.00
信息:  SG16    9 -> 8309.00
信息:  =====================================
信息:  --- END OF REPORT ----
CALL
```

PL/SQL Processing with Cursors: Sample (In Oracle)

FIGURE 8.32 A SIMPLE PRC_CURSOR EXAMPLE

SQL Plus

```
SQL> CREATE OR REPLACE PROCEDURE PRC_CURSOR_EXAMPLE IS
  2  W_P_CODE    PRODUCT.P_CODE%TYPE;
  3  W_P_DESCRIPT        PRODUCT.P_DESCRIPT%TYPE;
  4  W_TOT               NUMBER(3);
  5  CURSOR PROD_CURSOR IS
  6     SELECT P_CODE, P_DESCRIPT
  7        FROM PRODUCT
  8        WHERE P_QOH > (SELECT AVG(P_QOH) FROM PRODUCT);
  9  BEGIN
 10  DBMS_OUTPUT.PUT_LINE('PRODUCTS WITH P_QOH > AVG(P_QOH)');
 11  DBMS_OUTPUT.PUT_LINE('=====================================');
 12  OPEN PROD_CURSOR;
 13  LOOP
 14     FETCH PROD_CURSOR INTO W_P_CODE, W_P_DESCRIPT;
 15     EXIT WHEN PROD_CURSOR%NOTFOUND;
 16     DBMS_OUTPUT.PUT_LINE(W_P_CODE ||' -> ' || W_P_DESCRIPT );
 17  END LOOP;
 18  DBMS_OUTPUT.PUT_LINE('=====================================');
 19  DBMS_OUTPUT.PUT_LINE('TOTAL PRODUCT PROCESSED ' || PROD_CURSOR%ROWCOUNT);
 20  DBMS_OUTPUT.PUT_LINE('--- END OF REPORT ----');
 21  CLOSE PROD_CURSOR;
 22  END;
 23  /

Procedure created.

SQL> EXEC PRC_CURSOR_EXAMPLE;
PRODUCTS WITH P_QOH > AVG(P_QOH)
=====================================
PVC23DRT -> PVC pipe, 3.5-in., 8-ft
SM-18277 -> 1.25-in. metal screw, 25
SW-23116 -> 2.5-in. wd. screw, 50
=====================================
TOTAL PRODUCT PROCESSED 3
--- END OF REPORT ----

PL/SQL procedure successfully completed.

SQL> _
```

Triggers (触发器)

自动启用

- Procedural SQL code automatically invoked by RDBMS when given data manipulation event occurs
- Parts of a trigger definition
  - Triggering timing: indicates when trigger's PL/SQL code executes
  - Triggering event: statement that causes the trigger to execute
    - Triggering level: statement- and row-level
    - Triggering action: PL/SQL code enclosed between the BEGIN and END keywords

- DROP TRIGGER trigger_name command
  - Deletes a trigger without deleting the table
- Trigger action based on conditional DML predicates
  - Actions depend on the type of DML statement that fires the trigger

触发器就是特殊的存储过程

Triggers: Sample (In PG)

https://blog.csdn.net/Jon_Celoon/article/details/120080836?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_utm_term~default-4.pc_relevant_antiscanv2&spm=1001.2101.3001.4242.3&utm_relevant_index=7
触发器相关的CSDN教程

创建触发器以前，必须定义触发器使用的函数。这个函数不能有任何参数，它的返回值的类型必须是trigger。函数定义好以后，用命令CREATE TRIGGER创建触发器。多个触发器可以使用同一个函数。

y Editor

```
CREATE OR REPLACE FUNCTION emp_stamp() RETURNS trigger AS $emp_stamp$
    BEGIN
        -- Check that empname and salary are given
        IF NEW.empname IS NULL THEN
            RAISE EXCEPTION 'empname cannot be null';
        END IF;
        IF NEW.salary IS NULL THEN
            RAISE EXCEPTION '% cannot have null salary', NEW.empname;
        END IF;

        -- Who works for us when they must pay for it?
        IF NEW.salary < 0 THEN
            RAISE EXCEPTION '% cannot have a negative salary', NEW.empname;
        END IF;

        -- Remember who changed the payroll when
        NEW.last_date := current_timestamp;
        NEW.last_user := current_user;
        RETURN NEW;
    END;
$emp_stamp$ LANGUAGE plpgsql;

CREATE TRIGGER emp_stamp BEFORE INSERT OR UPDATE ON emp
FOR EACH ROW EXECUTE FUNCTION emp_stamp();
```

当一个pl函数作为一个触发器被调用时，系统自动在最外层的快创建一些特殊的变量，比如NEW,OLD...

NEW的数据类型是record，对于行级触发器，存有新的数据行

SQL的条件语句要有END IF

在SQL中，给变量赋值要用“ :="
=用于条件语句中的判断过程

FOR EACH ROW意思是它是个行级触发器

Output Explain Messages Notifications Query History

TE TRIGGER

触发器函数必须返回一个NULL或者一个记录类型的变量，这个变量的结构必须与触发器作用的表的结构一样

Query Editor

NOTICE：语句级的触发器应该返回NULL

```
CREATE TABLE emp (
    empname text,
    salary integer,
    last_date timestamp,
    last_user text
);
```

Query Editor

```
insert into emp values ('test', 1000);
select * from emp;
```

Data Output Explain Messages Notifications Query History

	empname text	salary integer	last_date timestamp without time zone	last_user text
1	test	1000	2020-03-23 23:45:29.516687	postgres

Query Editor

```
insert into emp values ('test', -1000);
```

Data Output Explain Messages Notifications Query Hi

```
ERROR:  错误:  test cannot have a negative salary
CONTEXT:  在RAISE的第13行的PL/pgSQL函数emp_stamp()

SQL state: P0001
```

可以用于用户进行更新、删除商品信息
还可以用于技术处理统计值

Triggers: Sample (In Oracle)

FIGURE 8.21 THE THIRD VERSION OF THE TRG_PRODUCT_REORDER TRIGGER

SQL Plus

```
SQL> CREATE OR REPLACE TRIGGER TRG_PRODUCT_REORDER
  2  BEFORE INSERT OR UPDATE OF P_QOH, P_MIN ON PRODUCT
  3  FOR EACH ROW
  4  BEGIN
  5     IF :NEW.P_QOH <= :NEW.P_MIN THEN
  6        :NEW.P_REORDER := 1;
  7     ELSE
  8        :NEW.P_REORDER := 0;
  9     END IF;
 10  END;
 11  /

Trigger created.

SQL> _
```

Triggers: Sample (In Oracle)

FIGURE 8.22 EXECUTION OF THE THIRD TRIGGER VERSION

SQL Plus

```
SQL> SELECT P_CODE, P_DESCRIPT, P_QOH, P_MIN, P_MIN_ORDER, P_REORDER
  2  FROM PRODUCT;

P_CODE   P_DESCRIPT                          P_QOH P_MIN P_MIN_ORDER P_REORDER
-------- ---------------------------------- ----- ----- ----------- ---------
11QER/31 Power painter, 15 psi., 3-nozzle      29     5          25         1
13-Q2/P2 7.25-in. pwr. saw blade               32    15          50         0
14-Q1/L3 9.00-in. pwr. saw blade               18    12          50         0
1546-QQ2 Hrd. cloth, 1/4-in., 2x50             15     8          35         0
1558-QW1 Hrd. cloth, 1/2-in., 3x50             23     5          25         0
2232/QTY B&D jigsaw, 12-in. blade               8     5          15         0
2232/QWE B&D jigsaw, 8-in. blade                6     7          15         1
2238/QPD B&D cordless drill, 1/2-in.           12     5          12         0
23109-HB Claw hammer                           23    10          25         0
23114-AA Sledge hammer, 12 lb.                  8    10          12         1
54778-2T Rat-tail file, 1/8-in. fine           43    20          25         0
89-WRE-Q Hicut chain saw, 16 in.               11     5          10         0
PVC23DRT PVC pipe, 3.5-in., 8-ft              188    75          50         0
SM-18277 1.25-in. metal screw, 25             172    75          50         0
SW-23116 2.5-in. wd. screw, 50                237   100         100         0
WR3/TT3  Steel matting, 4'x8'x1/6", .5" mesh   18     5          10         0

16 rows selected.

SQL> UPDATE PRODUCT SET P_QOH = P_QOH;

16 rows updated.

SQL> SELECT P_CODE, P_DESCRIPT, P_QOH, P_MIN, P_MIN_ORDER, P_REORDER
  2  FROM PRODUCT
  3  WHERE P_CODE = '11QER/31';

P_CODE   P_DESCRIPT                          P_QOH P_MIN P_MIN_ORDER P_REORDER
-------- ---------------------------------- ----- ----- ----------- ---------
11QER/31 Power painter, 15 psi., 3-nozzle      29     5          25         0

SQL> _
```

Embedded SQL

- SQL statements contained within an application programming language
  - Host language: any language that contains embedded SQL statements
- Differences between SQL and procedural languages
  - Run-time mismatch
    - SQL is executed one instruction at a time
    - Host language runs at client side in its own memory space
  - Processing mismatch
    - Conventional programming languages process one data element at a time
    - Newer programming environments manipulate data sets in a cohesive manner
  - Data type mismatch
    - Data types provided by SQL might not match data types used in different host languages

Embedded SQL (cont')

- Embedded SQL framework defines:
  - Standard syntax to identify embedded SQL code within the host language
  - Standard syntax to identify host variables
  - Communication area used to exchange status and error information between SQL and host language
- Static SQL: programmer uses predefined SQL statements and parameters 静态SQL
  - SQL statements will not change while application is running
- Dynamic SQL: SQL statement is generated at run time
  - Attribute list and condition are not known until end user specifies them
  - Slower than static SQL
  - Requires more computer resources
  - Inconsistent levels of support and incompatibilities among DBMS vendors

```
EXEC SQL
        SQL statement;
END-EXEC.
```

Extensions in PostgreSQL

Query Editor

```
CREATE EXTENSION plpython3u;
```

这个的意思是说

Data Output | Explain | Messages | Notifica

```
CREATE EXTENSION
```

Query Editor

```
CREATE OR REPLACE FUNCTION pymax (a integer, b integer)
RETURNS integer
AS $$
if a > b:
    return a
return b
$$ LANGUAGE plpython3u;
```

意思就是用python3写了一个大小比较器
还可以用来写计算器、调用文件什么的

```
SELECT pymax(110, 111);
```

Data Output | Explain | Messages | Notifica

	pymax integer
1	111

The PL/Python Extensions in PostgreSQL

Query Editor

```
1  CREATE OR REPLACE FUNCTION list_incoming_files()
2  RETURNS SETOF text AS
3  $$
4  import os
5  return os.listdir('d:/')
6  $$
7  LANGUAGE plpython3u;
```

Data Output | Explain | Messages | Notifications | Query History

```
CREATE FUNCTION

Query returned successfully in 123 msec.
```

Query Editor

```
1  SELECT list_incoming_files();
```

Data Output | Explain | Messages | Notif

	list_incoming_files text
5	ChromeSetup.exe
6	config.ini
7	Config.Msi

Database Concepts (V)

Database Connectivity

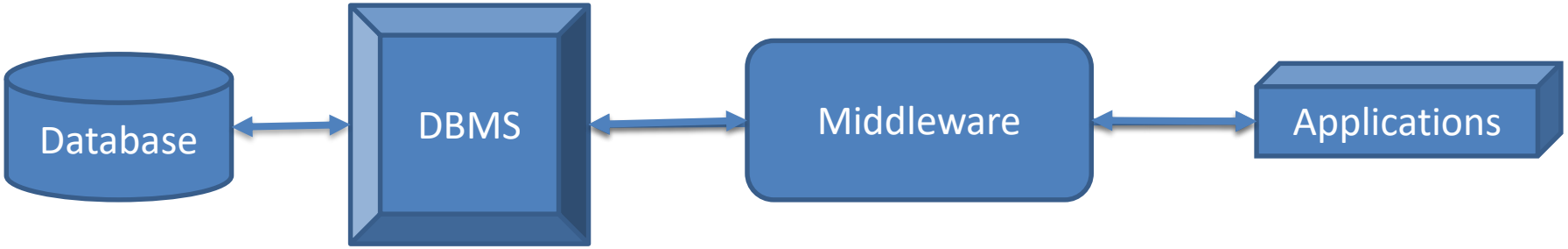
Chaokun Wang

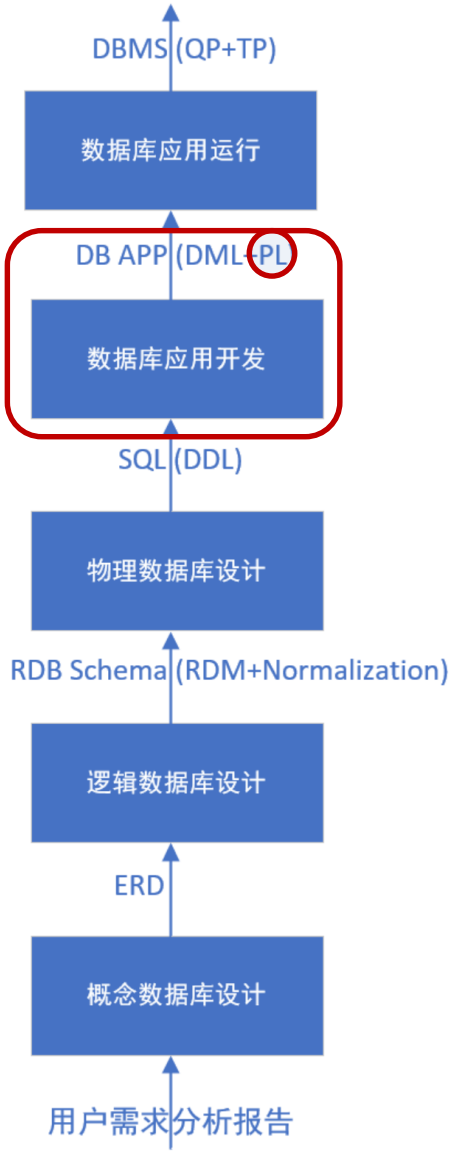
School of Software, Tsinghua University

chaokun@tsinghua.edu.cn

April 26, 2022

Motivation





Outline

- Database Connectivity Foundation
- PostgreSQL C Connector
- Introduction to Python

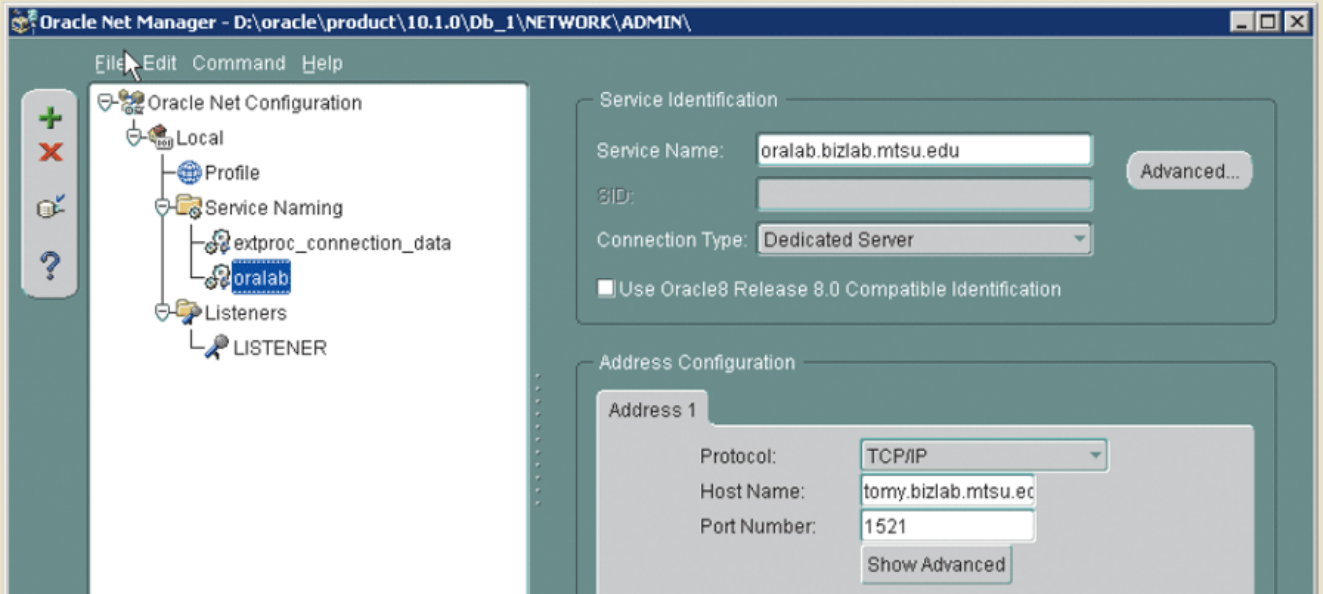
Database Connectivity

- Mechanisms through which application programs connect and communicate with data repositories
  - Database middleware: provides an interface between the application program and the database
  - Data repository: data management application used to store data generated by an application program
  - Universal Data Access (UDA): collection of technologies used to access any type of data source and manage the data through a common interface
    - ODBC, OLE-DB, and ADO.NET form the backbone of MS UDA architecture

Native SQL Connectivity

- Connection interface provided by database vendors, which is unique to each vendor
  - Interfaces are optimized for particular vendor's DBMS
  - Maintenance is a burden for the programmer

FIGURE 15.1 ORACLE NATIVE CONNECTIVITY



ODBC, DAO, and RDO (1 of 3)

- Open Database Connectivity (ODBC): Microsoft's implementation of a superset of SQL Access Group Call Level Interface (CLI) standard for database access
  - Widely supported database connectivity interface
  - Allows Windows application to access relational data sources by using SQL via standard application programming interface (API)
- Data Access Objects (DAO): object-oriented API used to access desktop databases such as MS Access and FileMaker Pro
  - Provides an optimized interface that expose functionality of Jet data engine to programmers

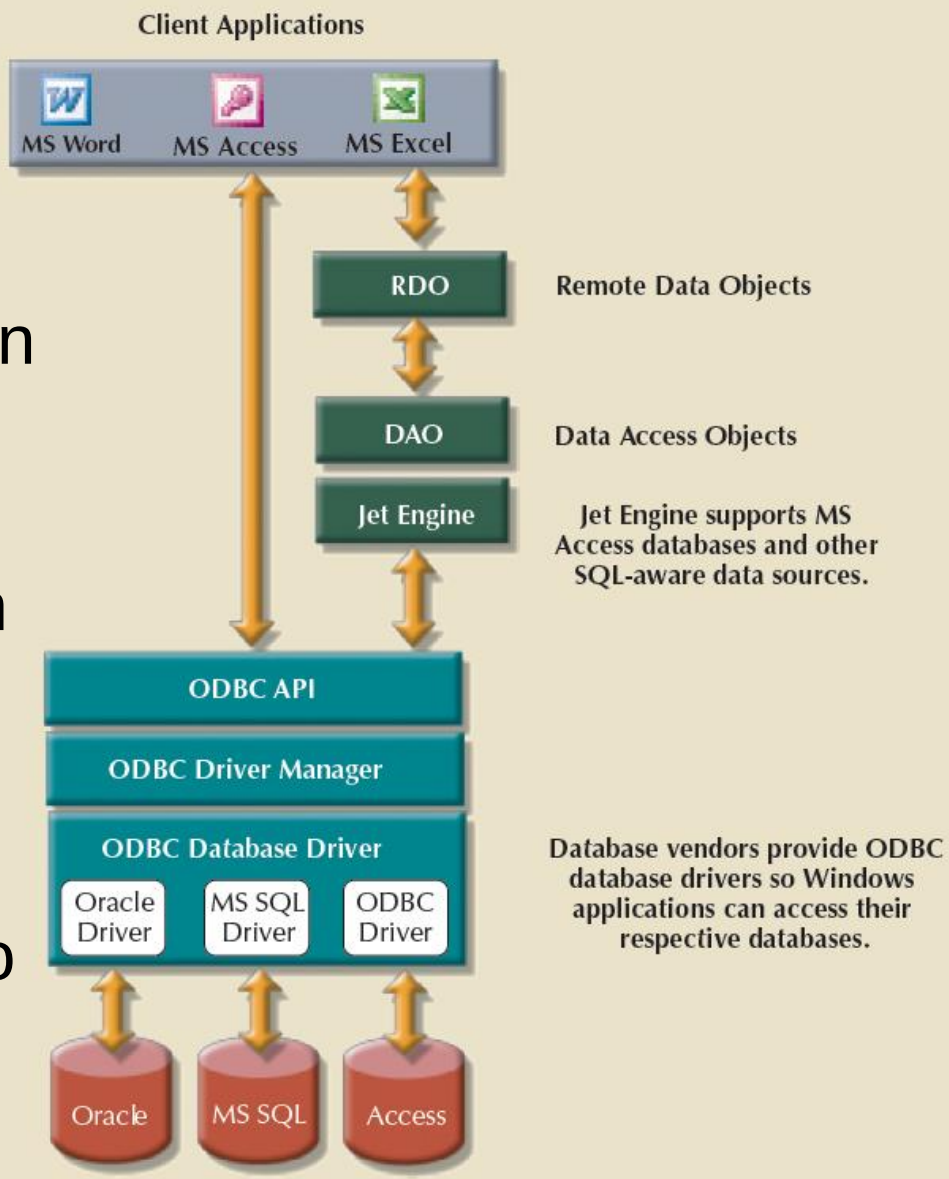
ODBC, DAO, and RDO (2 of 3)

- Remote Data Objects (RDO): higher-level object-oriented application interface used to access remote database servers
  - Optimized to deal with server-based databases
- Dynamic-link libraries (DLLs): implements ODBC, DAO, and RDO as shared code that is dynamically linked to the Windows operating environment

ODBC, DAO, and RDO (3 of 3)

- Components of ODBC architecture
  - High-level ODBC API through which application programs access ODBC functionality
  - Driver manager that is in charge of managing all database connections
  - ODBC driver that communicates directly to DBMS

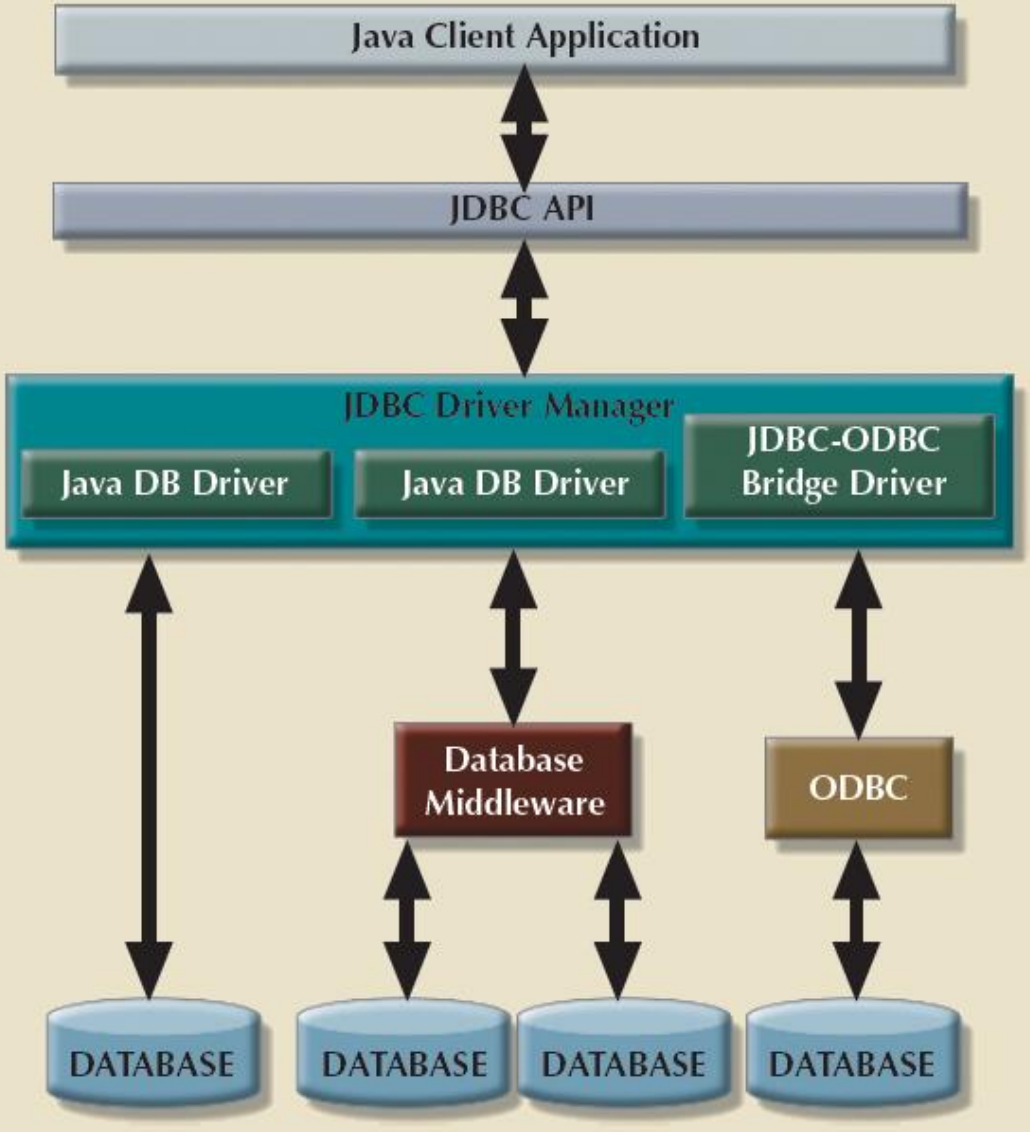
FIGURE 15.2 USING ODBC, DAO, AND RDO TO ACCESS DATABASES



Java Database Connectivity (JDBC)

- Application programming interface that allows a Java program to interact with a wide range of data sources
- Advantages of JDBC
  - Company can leverage existing technology and personnel training
  - Direct access to database server or access via database middleware
  - Programmers can use their SQL skills to manipulate the data in the company's databases
  - Provides a way to connect to databases through an ODBC driver

FIGURE 15.7 JDBC ARCHITECTURE

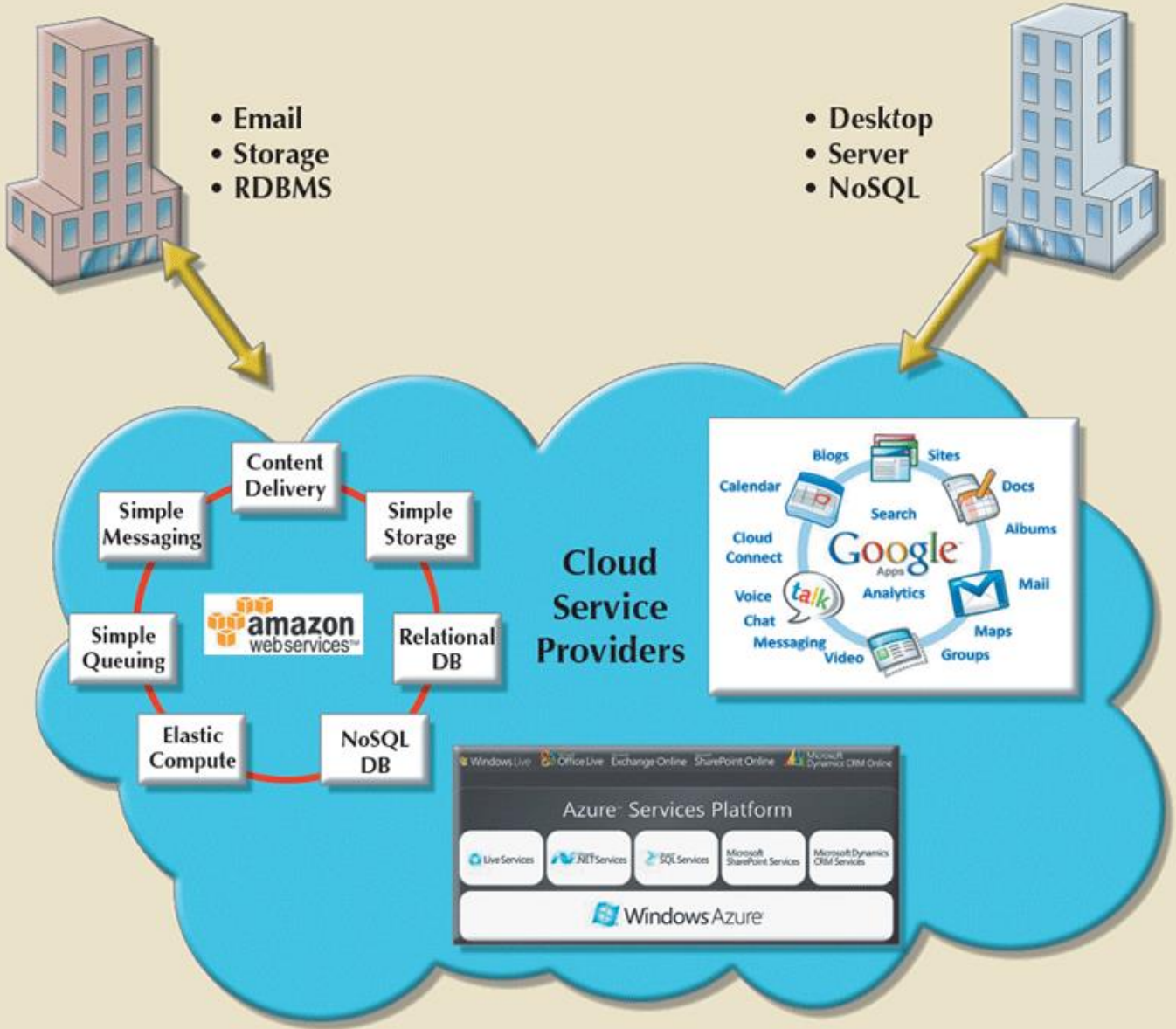


Cloud Computing Services

- Computing model that enables access to a shared pool of configurable computer resources
  - Can be rapidly provisioned and released with minimal management effort or service provider interaction
  - Potential to become a game changer; eliminates financial and technological barriers

Cloud Computing Services

FIGURE 15.21 CLOUD SERVICES

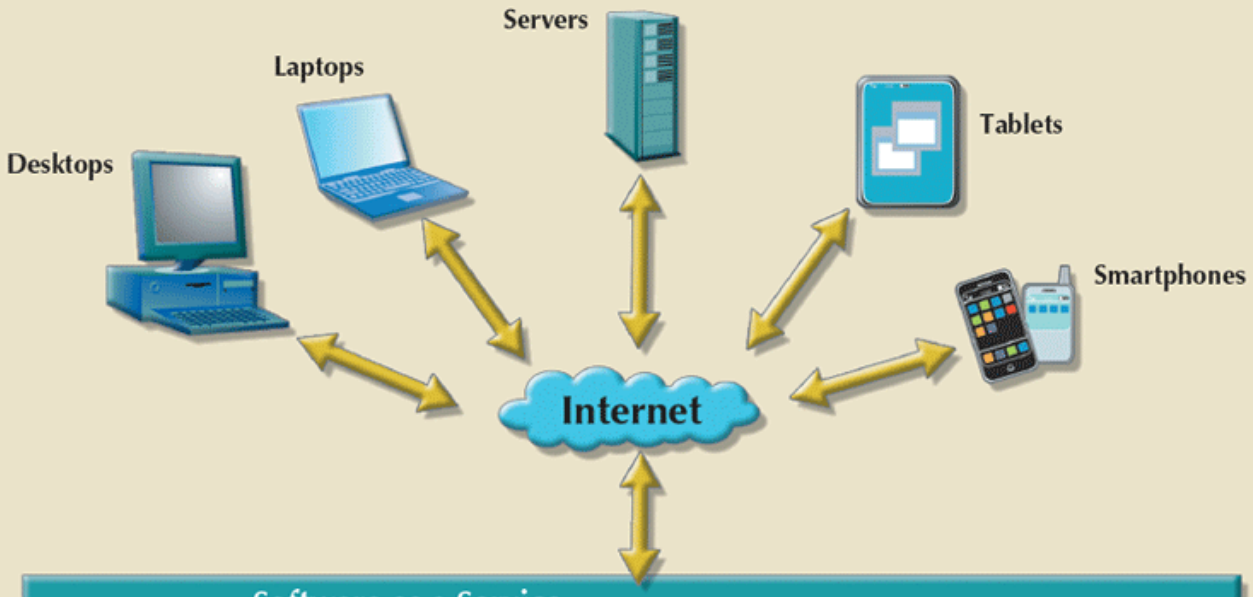


Cloud Implementation Types

- Public cloud
  - Built by a third-party organization to sell cloud services to the general public
- Private cloud
  - Built by an organization for the sole purpose of servicing its own needs
- Community cloud
  - Built by and for a specific group of organizations that share a common trade

Types of Cloud Services

FIGURE 15.23 TYPES OF CLOUD SERVICES



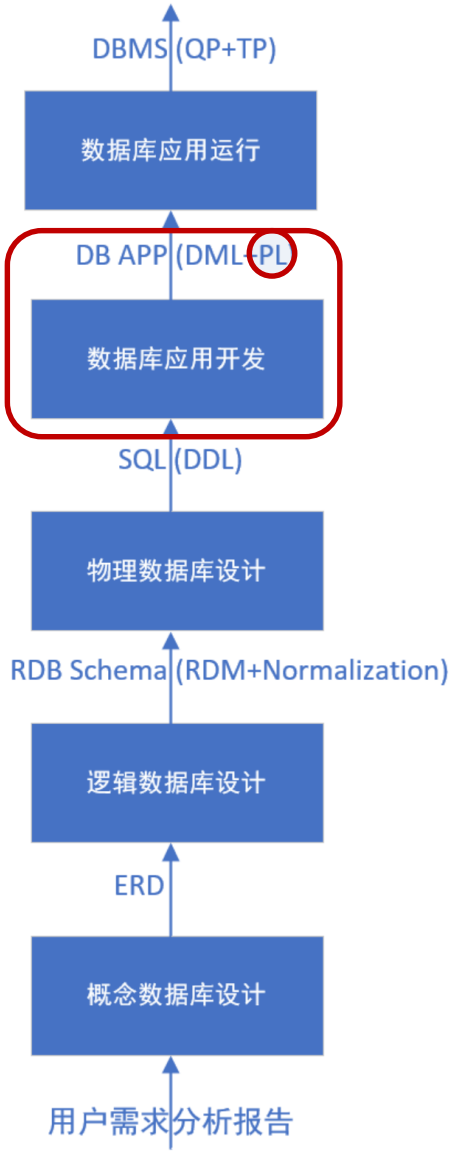
Software as a Service
- MS Office Live, MS Exchange Online
- Google Docs, Google Email
- Salesforce CRM Online
- SAP Business ByDesign

Platform as a Service
- Amazon Web Services, Amazon Relational Data Service, Amazon Simple DB
- MS Azure Platform, MS SQL Service
- Google Application Engine
- Google Spanner Relational Database Service

Infrastructure as a Service
- Amazon Web Services Elastic Computing Cloud 2 (EC2)
- Amazon Elastic MapReduce Service
- Amazon Simple Storage Service (S3)
- Amazon Elastic Load Balancing Service

Cloud Services: Advantages and Disadvantages

Table 15.4: Advantages and Disadvantages of Cloud Computing	
Advantage	**Disadvantage**
Low initial cost of entry. Cloud computing has lower costs of entry when compared with the alternative of building in house.	Issues of security, privacy, and compliance. Trusting sensitive company data to external entities is difficult for most data-cautious organizations.
Scalability/elasticity. It is easy to add and remove resources on demand.	Hidden costs of implementation and operation. It is hard to estimate bandwidth and data migration costs.
Support for mobile computing. Cloud computing providers support multiple types of mobile computing devices.	Data migration is a difficult and lengthy process. Migrating large amounts of data to and from the cloud infrastructure can be difficult and time-consuming.
Ubiquitous access. Consumers can access the cloud resources from anywhere at any time, as long as they have Internet access.	Complex licensing schemes. Organizations that implement cloud services are faced with complex licensing schemes and complicated service-level agreements.
High reliability and performance. Cloud providers build solid infrastructures that otherwise are difficult for the average organization to leverage.	Loss of ownership and control. Companies that use cloud services are no longer in complete control of their data. What is the responsibility of the cloud provider if data are breached? Can the vendor use your data without your consent?
Fast provisioning. Resources can be provisioned on demand in a matter of minutes with minimal effort.	Organization culture. End users tend to be resistant to change. Do the savings justify being dependent on a single provider? Will the cloud provider be around in 10 years?
Managed infrastructure. Most cloud implementations are managed by dedicated internal or external staff. This allows the organization's IT staff to focus on other areas.	Difficult integration with internal IT system. Configuring the cloud services to integrate transparently with internal authentication and other internal services could be a daunting task.



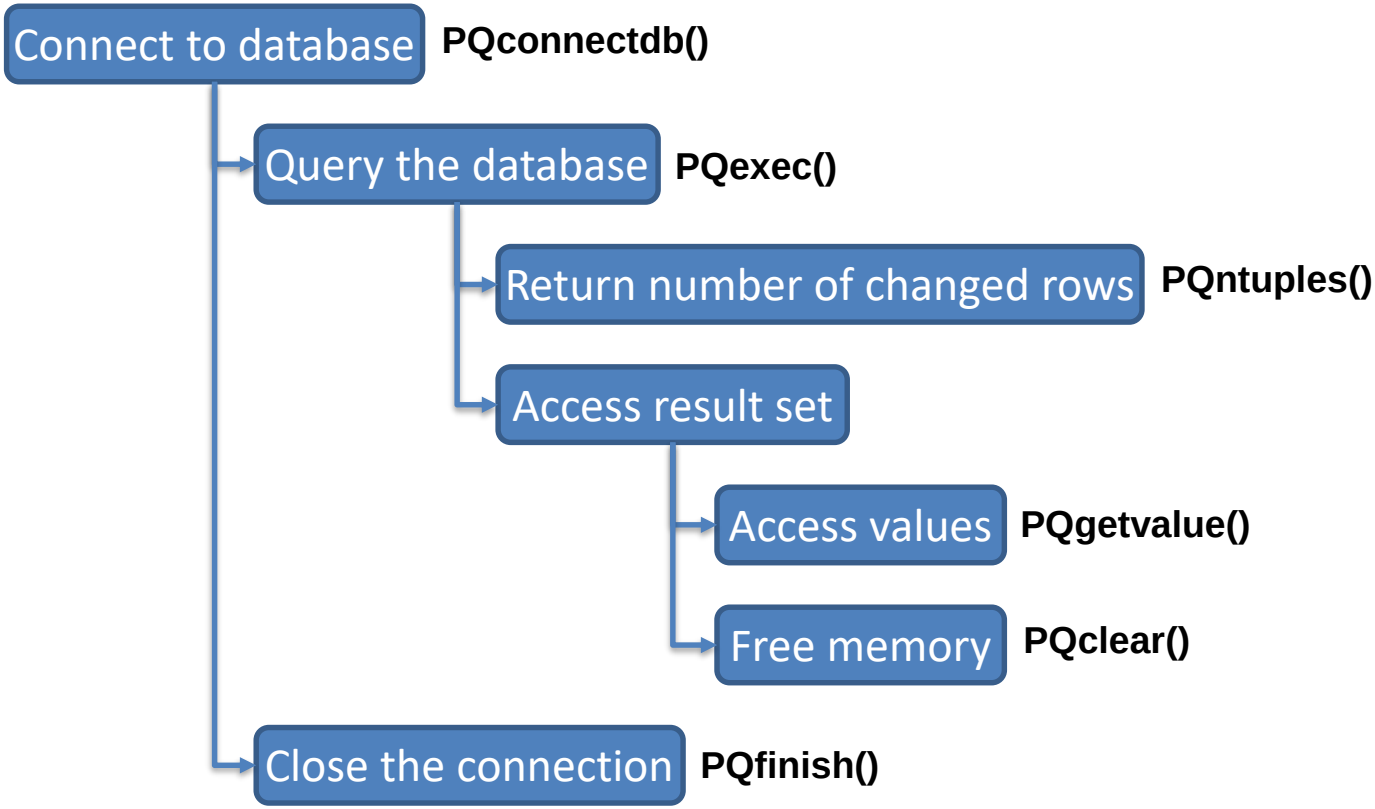
Outline

- Database Connectivity Foundation
- PostgreSQL C Connector
- Introduction to Python

Tutorial: PostgreSQL C Connector

- Pre-requirement
  - PostgreSQL Connector/C
    - x86 vs. x64
    - Can be found in PostgreSQL installation directory
  - C IDE & Compiler
    - Eclipse CDT + MinGW or Microsoft Visual Studio
    - Include “PgInstallationDir\include” in the include path
    - Add “PgInstallationDir\lib\libpq.lib” into the linker library
    - Include “PgInstallationDir\bin” to the system path

Tutorial: PostgreSQL C Connector



Please refer to https://www.postgresql.org/docs/14/libpq.html

Conclusions

- Stored procedures
- Triggers
- PL/SQL functions
- Extensions in PostgreSQL
  - PL/Python
- Database Connectivity Foundation
  - Native SQL connectivity (vendor provided)
  - ODBC/DAO/RDO/JDBC
  - Cloud Computing Services
- PostgreSQL C Connector

Homework

- Read the following Chapters of DS1

  § 8.7 (pp 401-418)

  § 8.8 (pp 419-423)

  § 15.1 (pp 693-704)

  § 15.4 (pp 722-729)

- Assignment
  - Later in Xuetang

- Further Reading

  § 15.2-15.3 of DS1

Happy 5.1 Holiday!

Thank you!