SQL简介及基本语法

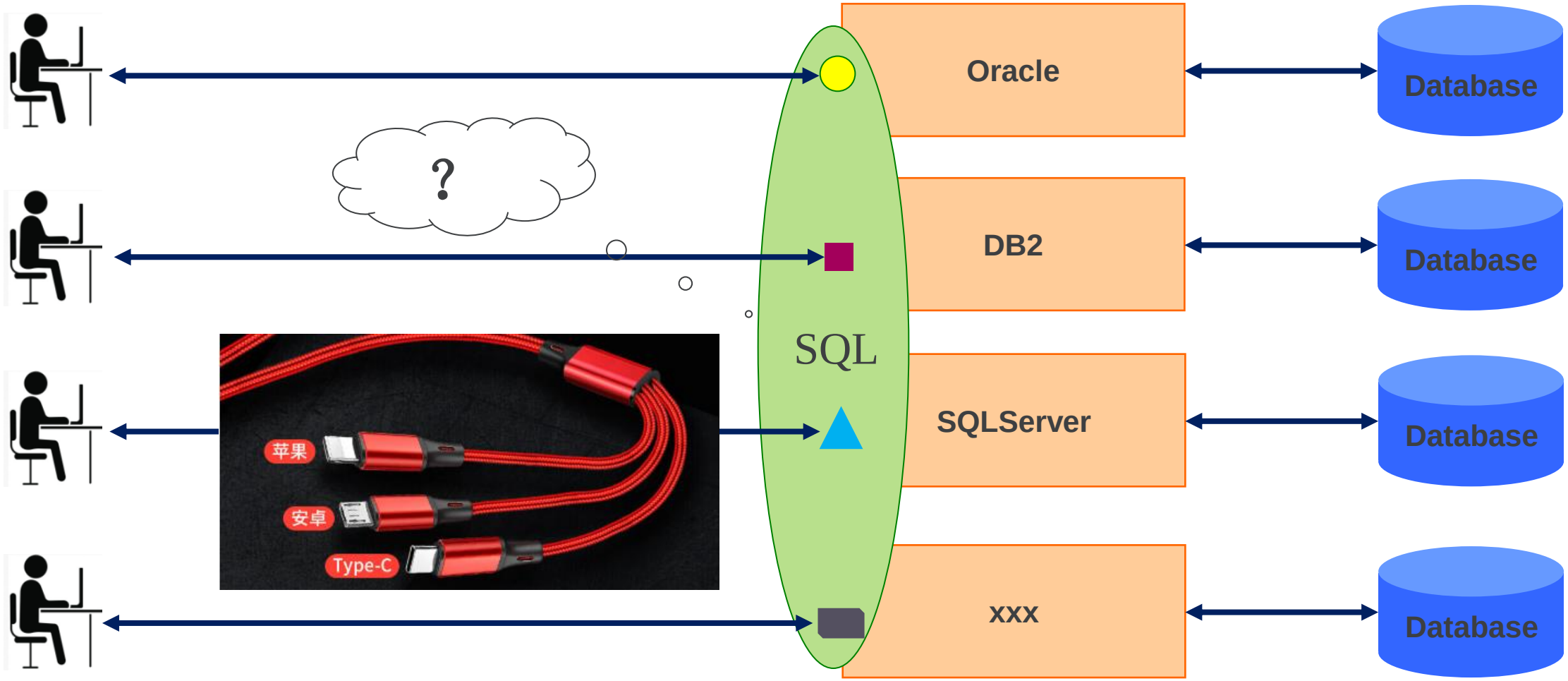
讲解人：李鸿岐

SQL简介

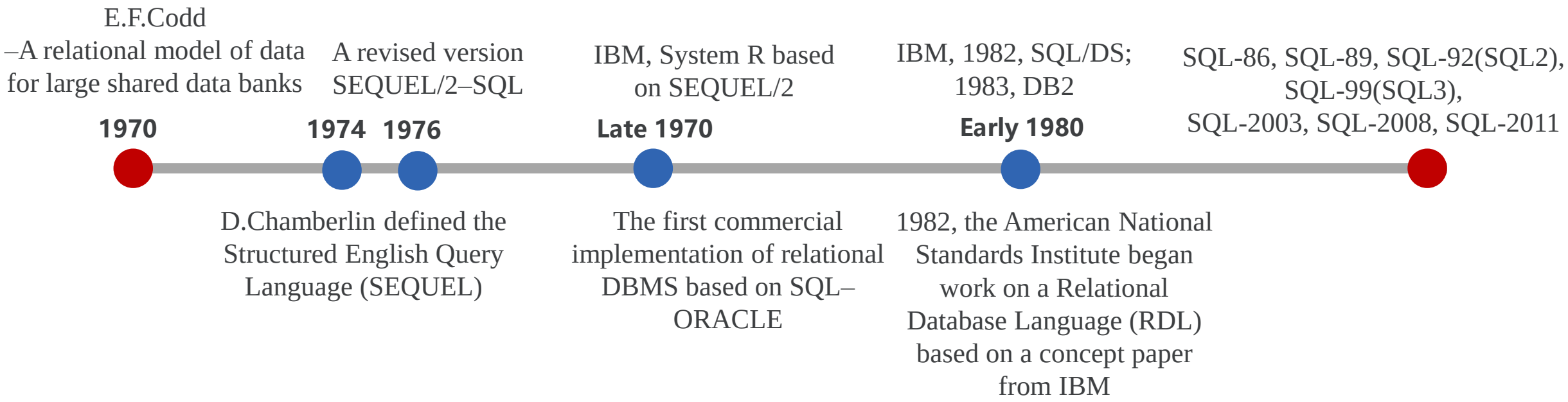
结构化查询语言（SQL）

- What is SQL
  - SQL is abbreviation of **Structured Query Language**.
  - It is the way we access the database.
- Objectives of SQL: Allow a user to
  - Create the database and relation structure.
  - Perform basic data management tasks.
  - Perform both simple and complex queries.
  - Perform these tasks with minimal user effort.
- SQL是关系数据库的标准语言
  - 是一个通用的、功能极强的关系数据库语言。

SQL起源


?
苹果
安卓
Type-C
SQL
Oracle
DB2
SQLServer
xxx
Database
Database
Database
Database

SQL发展


E.F.Codd
–A relational model of data for large shared data banks
1970
A revised version SEQUEL/2–SQL
1974 1976
D.Chamberlin defined the Structured English Query Language (SEQUEL)
IBM, System R based on SEQUEL/2
Late 1970
The first commercial implementation of relational DBMS based on SQL–ORACLE
IBM, 1982, SQL/DS; 1983, DB2
Early 1980
1982, the American National Standards Institute began work on a Relational Database Language (RDL) based on a concept paper from IBM
SQL-86, SQL-89, SQL-92(SQL2), SQL-99(SQL3), SQL-2003, SQL-2008, SQL-2011

SQL发展

标准	大致页数	发布日期
SQL/86		**1986.10**
SQL/89（FIPS 127-1）	**120页**	**1989年**
SQL/92	**622页**	**1992年**
SQL99（SQL 3）	**1700页**	**1999年**
SQL2003	**3600页**	**2003年**
SQL2008	**3777页**	**2006年**
SQL2011		**2010年**

- 目前，没有一个数据库系统能够支持SQL标准的所有概念和特性
  - 使用具体系统时要查阅各产品的用户手册（User Manuals）。

SQL特点

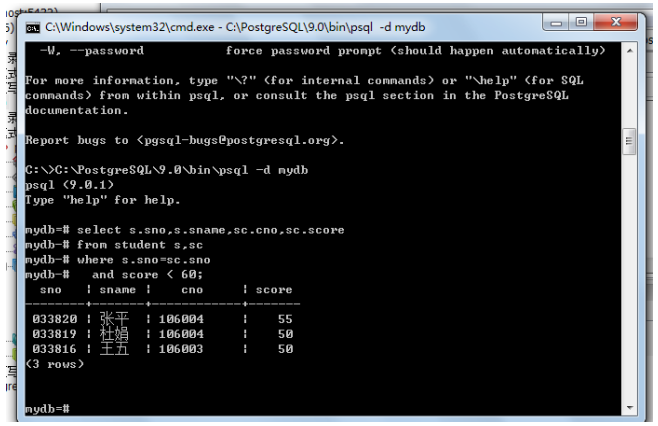
- **Non-procedual** - based on relational algebra and relational calculus.
- Easy to learn

SQL 功能	动词
Data query	SELECT
Data define	CREAT DEOP ALTER
Data manipulation	INSERT UPDATE DELETE
Data control	GRANT REVOKE

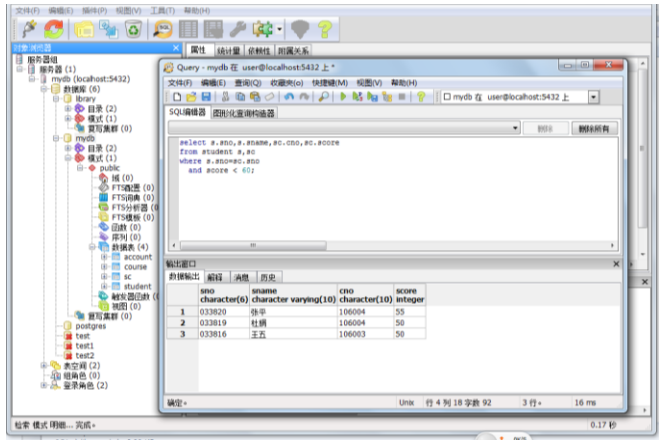
- 集数据定义语言（DDL），数据操纵语言（DML），数据控制语言（DCL）功能于一体
  - SQL只要提出“做什么”，无须了解存取路径
  - 存取路径的选择以及SQL的操作过程由系统自动完成。

SQL应用

- Use SQL interactively by entering the statements at a terminal.
- Embed SQL statements in a procedural language (programmatic SQL).



命令行



GUI

```
xxx xxxx ( ) {
    ……
    SQL access to dbms for retrieving data
    ……
}
```

嵌入高级程序语言

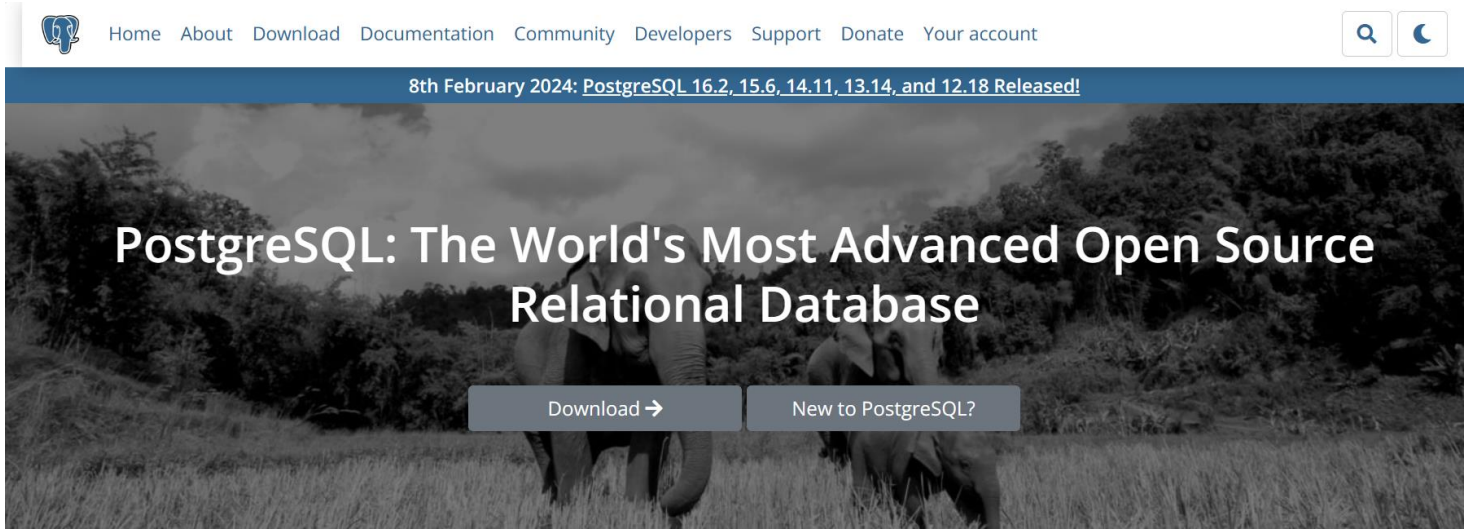
下载及安装

PostgreSQL下载及安装

选择正确版本？

https://zhuanlan.zhihu.com/p/646870620

https://www.postgresql.org/



New to PostgreSQL?

PostgreSQL is a powerful, open source object-relational database system with over 35 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.

There is a wealth of information to be found describing how to install and use PostgreSQL through the official documentation. The open source community provides many helpful places to become familiar with PostgreSQL, discover how it works, and find career opportunities. Learn more on how to engage with the community.

Latest Releases

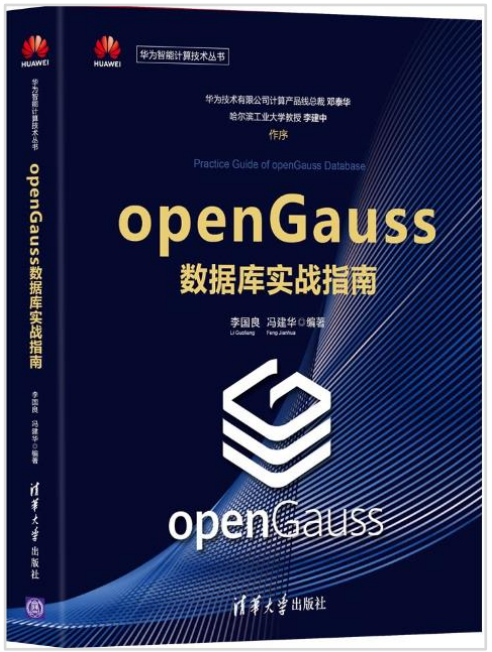
2024-02-08 - PostgreSQL 16.2, 15.6, 14.11, 13.14, and 12.18 Released!

The PostgreSQL Global Development Group has released an update to all supported versions of PostgreSQL, including 16.2, 15.6, 14.11, 13.14, and 12.18. This release fixes one security vulnerabilities and over 65 bugs reported over the last several months.

If you use GIN indexes, you may need to reindex after updating to this release. Please see the release notes for more information.

For the more information about this release, please review the release notes. You can download PostgreSQL from the download page.

16.2 · 2024-02-08 · Notes
15.6 · 2024-02-08 · Notes
14.11 · 2024-02-08 · Notes
13.14 · 2024-02-08 · Notes
12.18 · 2024-02-08 · Notes



《数据库》课程方案介绍.pptx

参考资料包网址.txt

鲲鹏计算实验资源删除指导手册.pptx

实验1：在ECS上安装部署openGauss数据库指导手册.docx

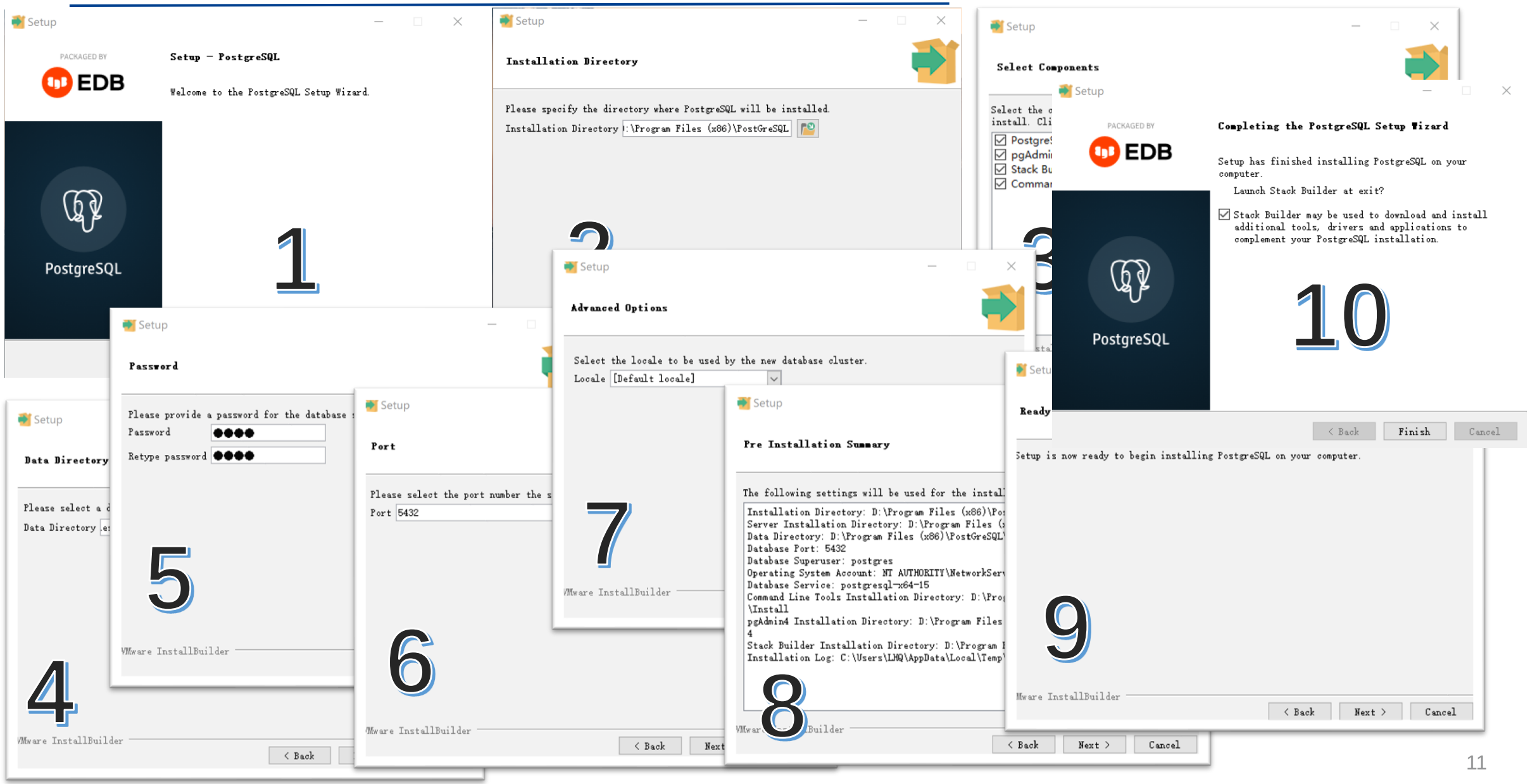
实验2：openGauss场景化综合应用实验.docx

实验3：GaussDB(for openGauss)多用户访问同一数据库实例指导手册.docx

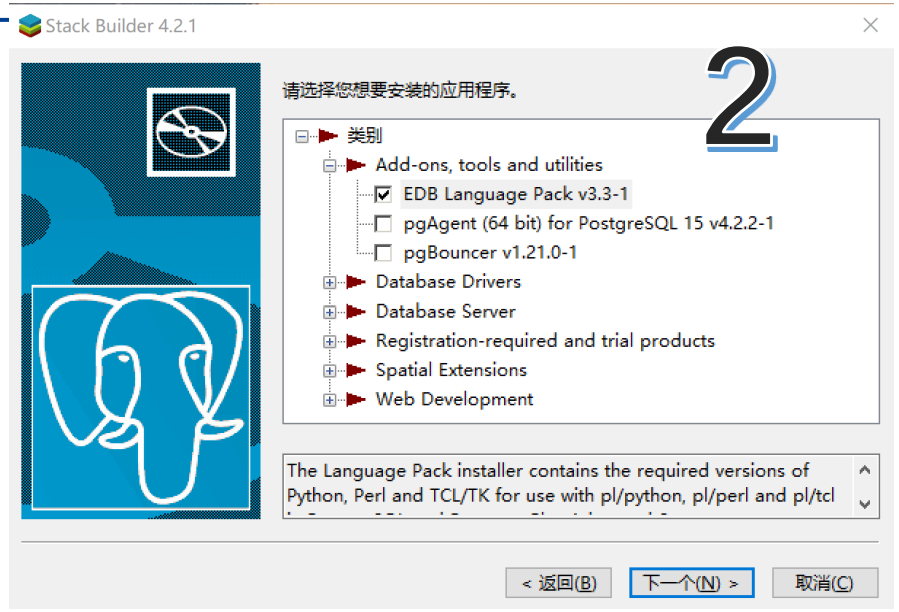
实验4：GaussDB(for openGauss)场景化综合应用实验（选作）.docx

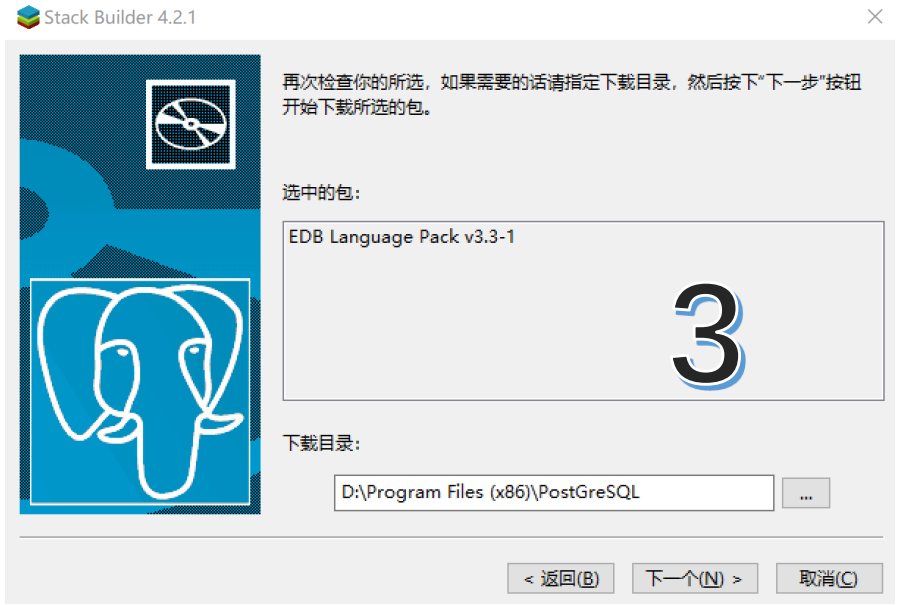
PostgreSQL下载及安装

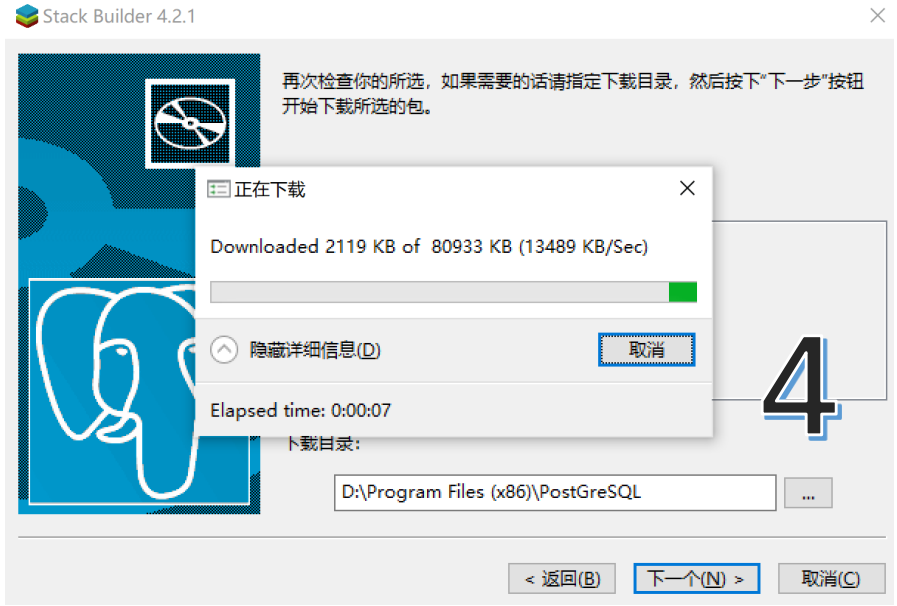
自定义安装目录时，建议不要选带空格的目录



PostgreSQL下载及安装



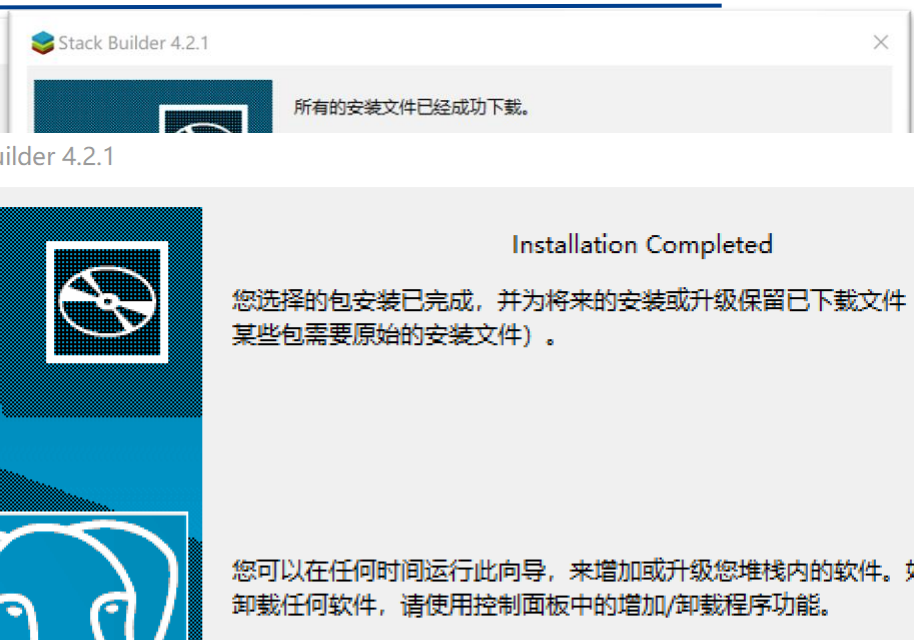


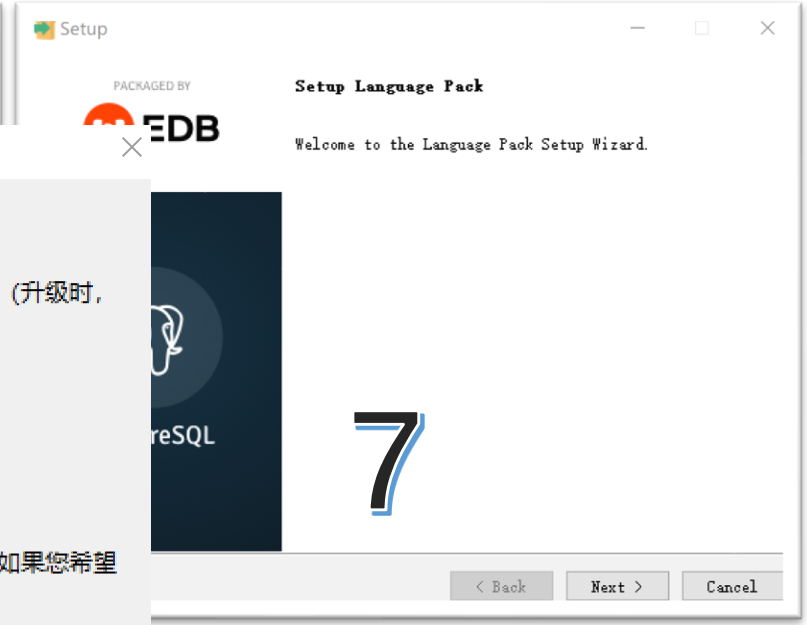


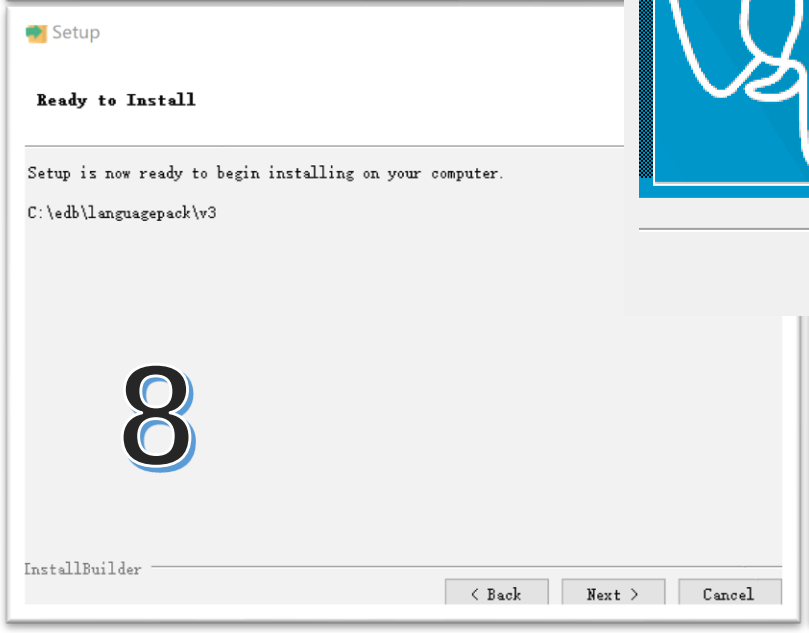
安装Stack Builder

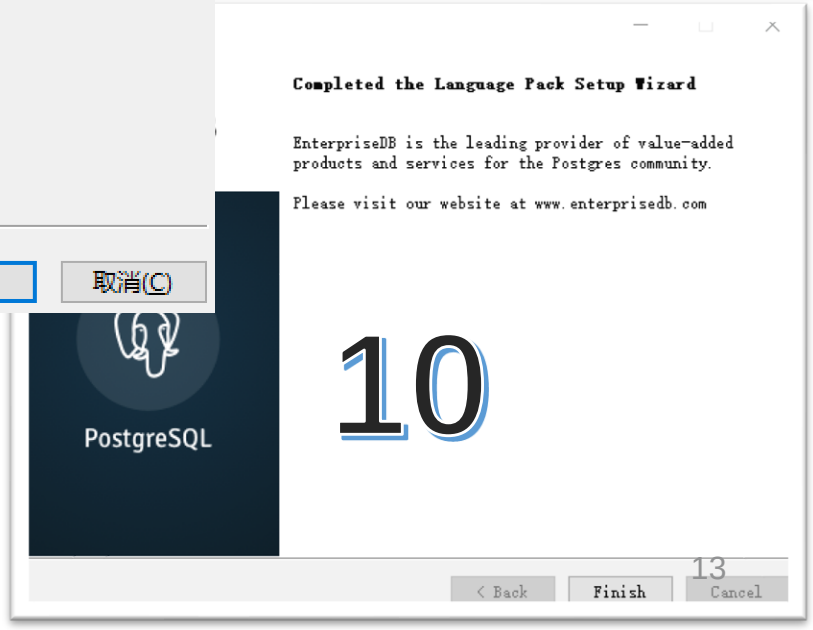
- 堆栈生成器；
- 封装用户数据，方便传递参数，封装后台数据库和文件的数据；
- 若仅仅是使用数据库的数据读写存储功能，可以不装。

PostgreSQL下载及安装

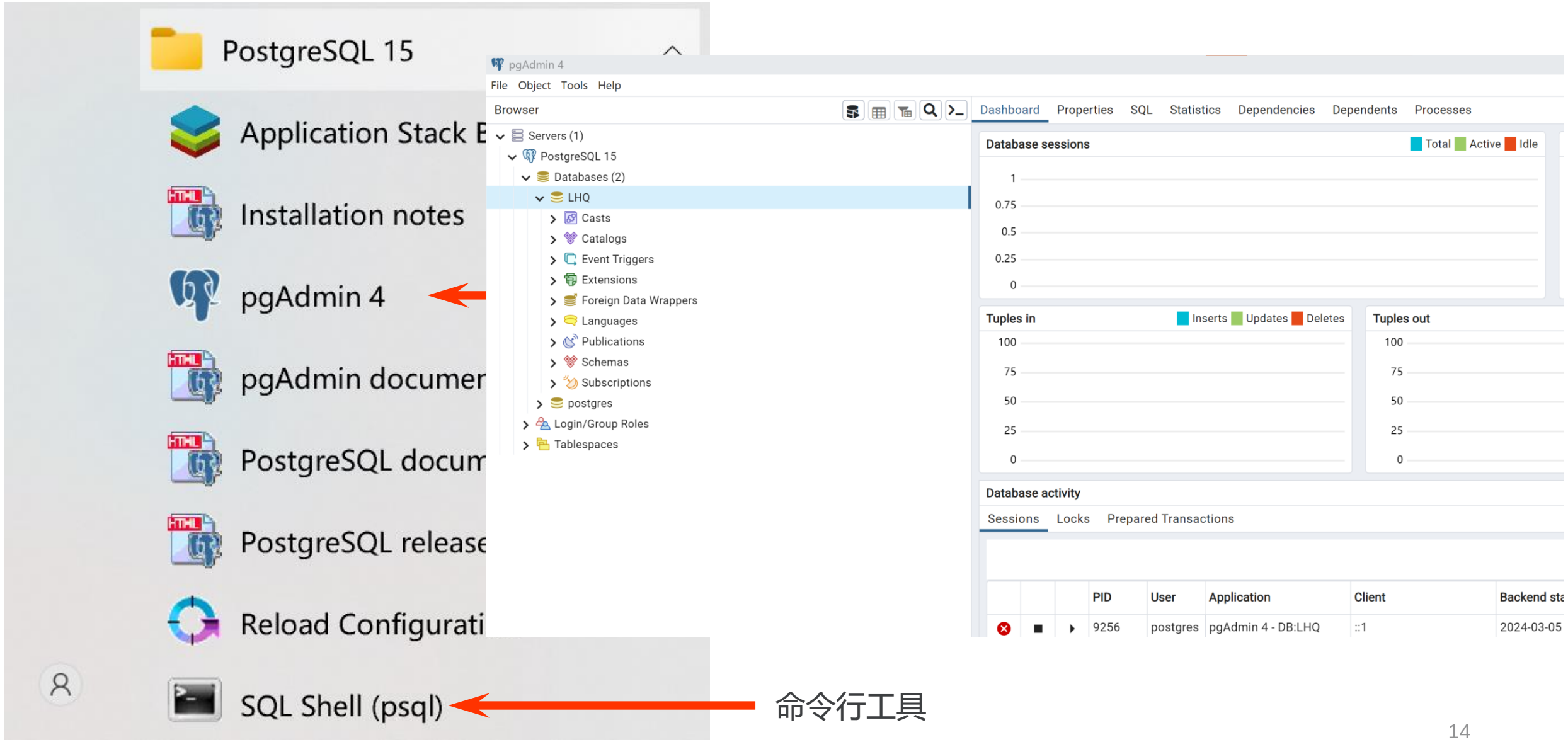






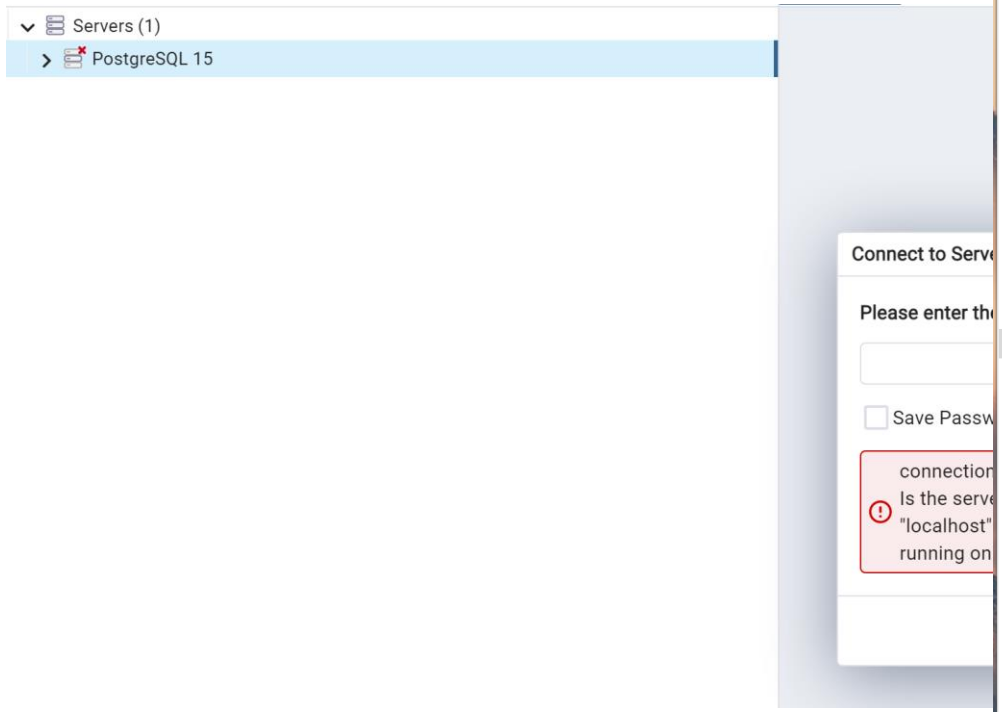


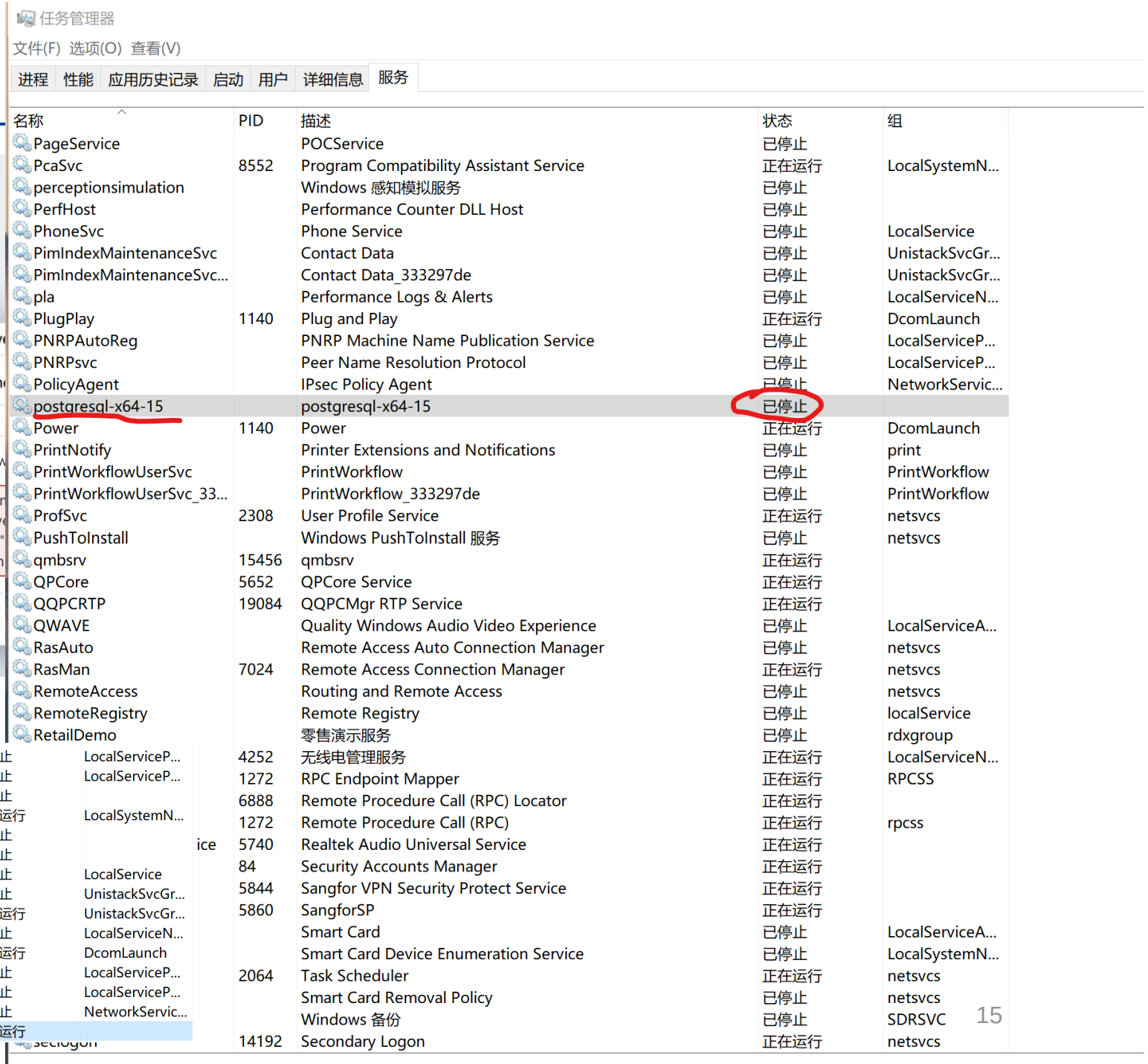
PostgreSQL下载及安装



命令行工具

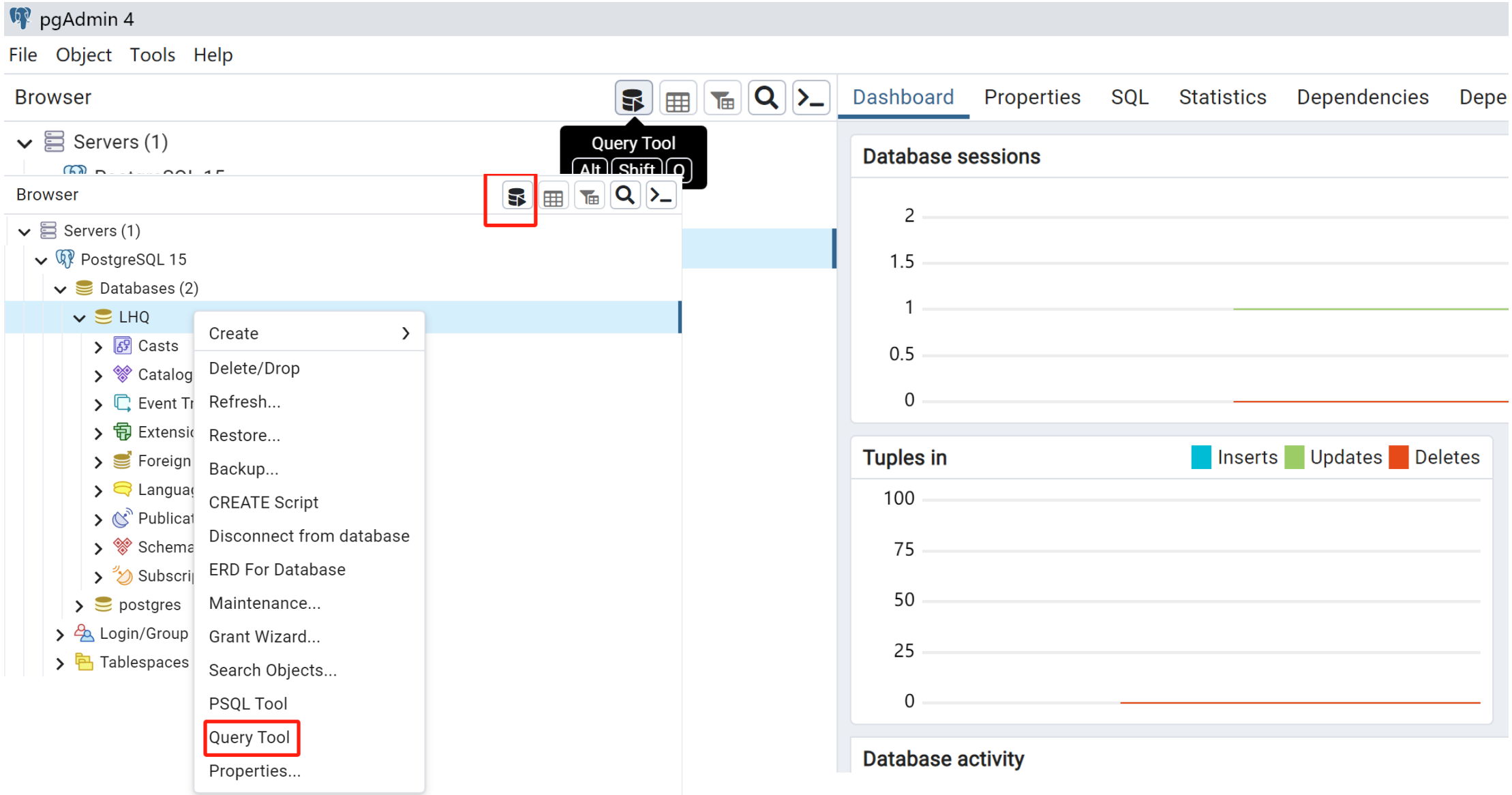
PostgreSQL下载及安装





p2pimsvc		Peer Networking Identity Manager	已停止	LocalServiceP...	
p2psvc		Peer Networking Grouping	已停止	LocalServiceP...	
PageService		POCService	已停止		
PcaSvc	8552	Program Compatibility Assistant Service	正在运行	LocalSystemN...	
perceptionsimulation		Windows 感知模拟服务	已停止		
PerfHost		Performance Counter DLL Host	已停止		
PhoneSvc		Phone Service	已停止	LocalService	
PimIndexMaintenanceSvc		Contact Data	已停止	UnistackSvcGr...	
PimIndexMaintenanceSvc...	61968	Contact Data_34640828	正在运行	UnistackSvcGr...	
pla		Performance Logs & Alerts	已停止	LocalServiceN...	
PlugPlay	1140	Plug and Play	正在运行	DcomLaunch	
PNRPAutoReg		PNRP Machine Name Publication Service	已停止	LocalServiceP...	
PNRPsvc		Peer Name Resolution Protocol	已停止	LocalServiceP...	
PolicyAgent		IPsec Policy Agent	已停止	NetworkServic...	
postgresql-x64-15	30952	postgresql-x64-15	正在运行		

PostgreSQL下载及安装



基本SQL

SQL基本语法

- An SQL statement consists of reserved words and user-defined words.
- Most components of an SQL statement are case insensitive.
- Usually a statement terminator (;) is used to end each SQL statement. (Although the standard does not require it).
- Although SQL is free-format, an SQL statement or set of statements is more readable if indentation and lineation are used.
  - Each clause in a statement should begain on a new line.
  - The beginning of each clause should line up with the beginning of other clause.
  - If a clause has several parts, they should each appear on a separate line and be indented.
  - …

SQL基本语法

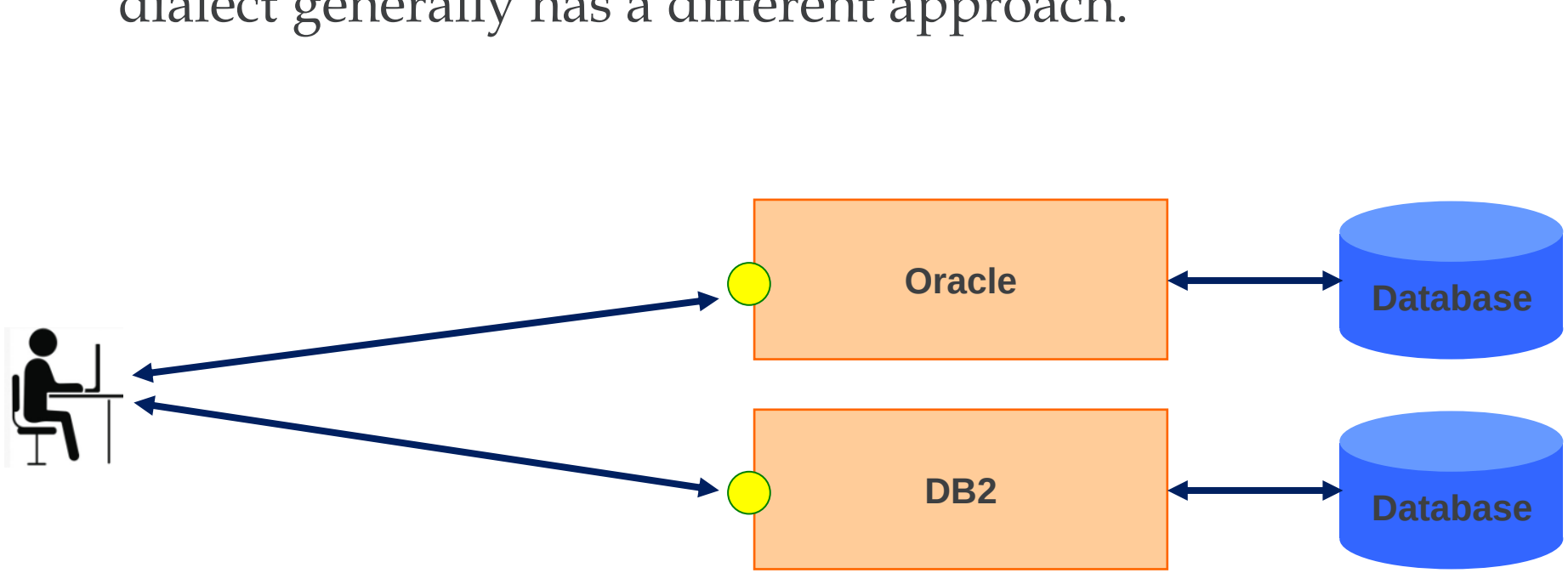
- SQL Identifiers.
  - Used to identify objects in the database, such as table names, view names, and columns
- SQL Scalar Data Types

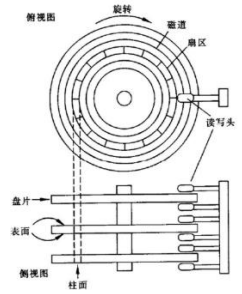
Data type	**Declarations**
Boolean	BOOLEAN
character	CHAR VARCHAR
bit	BIT BIT VARYING
exact numeric	NUMERIC DECIMAL INTEGER SMALLINT
approximate numeric	FLOAT REAL DOUBLE PRECISION
datetime	DATE TIME TIMESTAMP
interval	INTERVAL
large objects	CHARACTER LARGE OBJECT BINARY LARGE OBJECT

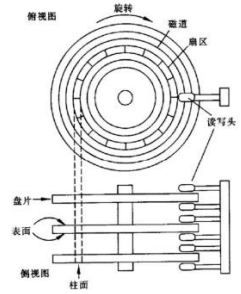
- Scalar operators

数据定义语言DDL

- Creating a Database.
  - The ISO standard does not specify how database are created, and each dialect generally has a different approach.







- 创建数据库涉及到存储空间的分配；
- 不同厂商策略不同；
- 参考具体DBMS的说明文档。

数据定义语言DDL

- Creating a table (CREATE TABLE)

CREATE TABLE TableName
{(columnName dataType [NOT NULL][UNIQUE]
[DEFAULT defaultOption][CHECK(searchCondition)][,…]}
[PRIMARY KEY (listOfColumns),] → 实体完整性约束
{[UNIQUE (listOfColumns),][,…]}
{[FOREIGN KEY (listOfForeignKeyColumns) → 参照完整性约束
REFERENCES ParentTableName[(listOfCandidateKeyColumns)],
[MATCH {PARTIAL | FULL}
[ON UPDATE referentialAction]
[ON DELETE referentialAction]][,…]}
{[CHECK(searchCondition)][,…]}) → 用户定义完整性约束

数据定义语言DDL

Creating a table (CREATE TABLE)

[案例] 建立“学生”表Student。学号是主码，姓名取值唯一。

```
CREATE TABLE Student
    (Sno   CHAR(9) PRIMARY KEY,
                          /* 列级完整性约束条件,Sno是主码*/
     Sname CHAR(20) UNIQUE,          /* Sname取唯一值*/
     Ssex   CHAR(2),
     Sage   SMALLINT,
     Sdept  CHAR(20)
    );
```

主码

UNIQUE 约束

数据定义语言DDL

Creating a table (CREATE TABLE)

[案例] 建立一个“课程”表Course

```
CREATE TABLE  Course
    (Cno      CHAR(4) PRIMARY KEY,
     Cname  CHAR(40),
     Cpno     CHAR(4),
     Ccredit  SMALLINT,
     FOREIGN KEY (Cpno) REFERENCES  Course(Cno)
    );
```

先修课

Cpno是外码
被参照表是Course
被参照列是Cno

数据定义语言DDL

Changing a Table Definition (ALTER TABLE)

ALTER TABLE TableName
[ADD [COLUMN] columnName dataType [NOT NULL][UNIQUE]
[DEFAULT defaultOption][CHECK(searchCondition)]]
[DROP [COLUMN] columnName [RESTRICT | CASCADE]]
[ADD [CONSTRAINT [constraintName]] tableConstraintDefinition]
[DROP CONSTRAINT constraintName [RESTRICT | CASCADE]]
[ALTER [COLUMN] SET DEFAULT defaultOption]
[ALTER [COLUMN] DROP DEFAULT]

Removing a Table (DROP TABLE)

DROP TABLE TableName [RESTRICT | CASCADE]

数据定义语言DDL

Changing a Table Definition (ALTER TABLE)

[案例] 向Student表增加“入学时间”列，其数据类型为日期型

```
ALTER TABLE Student ADD S_entrance DATE;
```

不管基本表中原来是否已有数据，新增加的列一律为空值

[案例] 将年龄的数据类型由字符型（假设原来的数据类型是字符型）改为整数。

```
ALTER TABLE Student ALTER COLUMN Sage INT;
```

[案例] 增加课程名称必须取唯一值的约束条件。

```
ALTER TABLE Course ADD UNIQUE(Cname);
```

数据定义语言DDL ◘ Removing a Table (DROP TABLE)

DROP TABLE TableName [RESTRICT | CASCADE]

- **RESTRICT**：删除表是有限制的。
  - 欲删除的基本表不能被其他表的约束所引用
  - 如果存在依赖该表的对象，则此表不能被删除
- **CASCADE**：删除该表没有限制。
  - 在删除基本表的同时，相关的依赖对象一起删除

[案例] 删除Student表

```
DROP TABLE  Student  CASCADE;
```

- 基本表定义被删除，数据被删除
- 表上建立的索引、视图、触发器等一般也将被删除

数据定义语言DDL

不同DBMS对数据定义语言DDL的处理不同

DROP TABLE时，SQL2011 与 3个RDBMS的处理策略比较

序号	标准及主流数据库的处理方式 / 依赖基本表的对象	SQL2011		Kingbase ES		Oracle 12c		MS SQL Server 2012
		R	C	R	C		C	
1	索引	无规定		√	√	√	√	√
2	视图	×	√	×	√	√ 保留	√ 保留	√ 保留
3	DEFAULT，PRIMARY KEY，CHECK（只含该表的列）NOT NULL 等约束	√	√	√	√	√	√	√
4	外码FOREIGN KEY	×	√	×	√	×	√	×
5	触发器TRIGGER	×	√	×	√	√	√	√
6	函数或存储过程	×	√	√ 保留	√ 保留	√ 保留	√ 保留	√ 保留

R表示RESTRICT , C表示CASCADE

'×'表示不能删除基本表，'√'表示能删除基本表，‘保留’表示删除基本表后，还保留依赖对象

案例

关系模式

department(dNo,dNa
student(sNo, sName,
course(cNo, cName, c
sc(sNo, cNo, score, re

参考脚本：

Create Tables.sql
Insert Values.sql

Create Tables.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
--DROP TABLE department CASCADE;
--DROP TABLE student CASCADE;
--DROP TABLE course CASCADE;
--DROP TABLE sc CASCADE;

--department(dNo,dName,officeRoom,homepa
--student(sNo,sName,sex,age,dNo)
--course(cNo,cName,cPNo,credit,dNo)
--sc(sNo,cNo,score,recordDate)

CREATE TABLE department(
  dNo          CHAR(2)       NOT NULL UNIQUE,
  dName        VARCHAR(20),
  officeRoom   VARCHAR(40),
  homePage     VARCHAR(80),
  PRIMARY KEY(dNo)
);

CREATE TABLE student(
  sNo                CHAR(6)       NOT NULL UN
  sName              VARCHAR(20)   NOT NULL
  sex                CHAR(2)       CHECK (sex IN
  age                INT,
  email        VARCHAR(50),
  dNo                CHAR(2),
  PRIMARY KEY(sNo),
  FOREIGN KEY (dNo) REFERENCES  departmen
);

CREATE TABLE course(
  cNo                CHAR(6)       NOT NULL U
  cName              VARCHAR(20)   NOT NULL
  cPNo               CHAR(6),
  credit     INT,
  dNo          CHAR(2),
  PRIMARY KEY(cNo),
  FOREIGN KEY (cPNo) REFERENCES  course(cN
  FOREIGN KEY (dNo)  REFERENCES  departmer
);
```

Insert Values.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
INSERT INTO department VALUES('01','信息学院','行政楼409','www.xxx.edu.cn');
INSERT INTO department VALUES('02','软件学院',null,null);
INSERT INTO department VALUES('03','理学院',null,null);
INSERT INTO department VALUES('04','文学院',null,null);
INSERT INTO department VALUES('05','外国语学院',null,null);

INSERT INTO student VALUES('170101','宁灿',  '女',19,'ningcan@mail.nwpu.edu.cn','01');
INSERT INTO student VALUES('170102','尹江月','女',19,null,'01');
INSERT INTO student VALUES('170103','杨佳伟','男',null,null,null);
INSERT INTO student VALUES('170104','杨何宇','男',19,null,'01');
INSERT INTO student VALUES('170105','胡耀斌','男',19,null,null);
INSERT INTO student VALUES('170106','李杨阳','女',20,null,'01');
INSERT INTO student VALUES('170107','杜利俊','女',18,null,'01');
INSERT INTO student VALUES('170108','钱多多','女',17,null,'01');
INSERT INTO student VALUES('170109','李佳伟','女',null,null,'01');
INSERT INTO student VALUES('170110','吴莫愁','女',21,null,'01');

INSERT INTO student VALUES('170201','安相成','男',19,null,'02');
INSERT INTO student VALUES('170202','曹师好','男',null,null,'02');
INSERT INTO student VALUES('170203','雷霆',  '男',18,null,'02');
INSERT INTO student VALUES('170204','刘书敏','男',20,null,'02');
INSERT INTO student VALUES('170205','王兵',  '男',21,null,'02');
INSERT INTO student VALUES('170206','李佳成','男',19,null,null);
INSERT INTO student VALUES('170207','唐玉迎','女',17,null,'02');
INSERT INTO student VALUES('170208','杨曼婷','女',19,null,'02');

INSERT INTO student VALUES('170301','张望',  '男',21,null,'03');
INSERT INTO student VALUES('170302','王芳',  '女',18,null,'03');
INSERT INTO student VALUES('170303','赵四海','男',19,null,'03');

INSERT INTO student VALUES('170401','孙敏',  '女',null,null,null);
INSERT INTO student VALUES('170402','李忠国','男',null,null,'04');
INSERT INTO student VALUES('170403','钱紧',  '男',17,null,'04');
INSERT INTO student VALUES('170404','钱多多','女',20,null,'04');
INSERT INTO student VALUES('170405','管八方','男',21,null,'04');
INSERT INTO student VALUES('170406','王兵',  '男',19,null,'04');
INSERT INTO student VALUES('170407','张三丰','男',100,null,null);

INSERT INTO course VALUES('030101','高等数学',null,2,'03');
INSERT INTO course VALUES('030102','线性代数',null,2,'03');
INSERT INTO course VALUES('030201','矩阵论','030102',3,'03');
```

数据查询SELECT

General form

SELECT [DISTINCT | ALL]{* | [columnExpression[AS newName]][,…]}
FROM TableName[alias][,…]
[**WHERE** condition]
[GROUP BY columnList][HAVING condition]
[ORDER BY columnList]

- **SELECT子句**：指定要显示的属性列
- **FROM子句**：指定查询对象（基本表或视图）
- **WHERE子句**：指定查询条件
- **GROUP BY子句**：对查询结果按指定列的值分组，该属性列值相等的元组为一个组。
- **HAVING短语**：只有满足指定条件的组才予以输出
- **ORDER BY子句**：对查询结果表按指定列值的升序或降序排序

数据查询SELECT

案例

SELECT *
FROM Student;

SELECT sNo,sName
FROM Student;

SELECT DISTINCT sNo
FROM SC;

SELECT sName, 2023-sAge
FROM Student;

```
288 select sname Name, 'Year of Birth:' Birth, 2023-age Birthday
289 from student;
```

Data Output | Messages | Notifications

	name character varying (20)	birth text	birthday integer
1	宁灿	Year of Birth:	2004
2	尹江月	Year of Birth:	2004
3	杨佳伟	Year of Birth:	[null]
4	杨何宇	Year of Birth:	2004
5	胡耀斌	Year of Birth:	2004

```
285 select sname, 'Year of Birth:', 2023-age
286 from student;
```

Data Output | Messages | Notifications

	sname character varying (20)	?column? text	?column? integer
1	宁灿	Year of Birth:	2004
2	尹江月	Year of Birth:	2004
3	杨佳伟	Year of Birth:	[null]
4	杨何宇	Year of Birth:	2004
5	胡耀斌	Year of Birth:	2004

数据查询SELECT

- Row selection (WHERE clause)

Condition	Predicates(谓词)
comparison	Comparison operators(=,>,<,<>,<=,>=,!=)
range	BETWEEN AND, NOT BETWEEN AND
set membership	IN, NOT IN
pattern match	LIKE, NOT LIKE
null	IS NULL, IS NOT NULL
logical operators	AND, OR,NOT

- 限制要查询的行，从关系中挑选一部分（满足条件的行）输出
  - Null 单独匹配（不能用传统的 =, !=）
  - 多个条件的逻辑表达式。

数据查询SELECT

案例

```
SELECT sName
FROM Student
WHERE sex = '男';
```

```
SELECT sNo,sName,dNo,age
FROM Student
WHERE age NOT BETWEEN 20 AND 23;
```

```
SELECT sName
FROM Student
WHERE age <= 17;
```

```
SELECT sName, sex
FROM Student
WHERE dNo NOT IN ('01', '02', '03');
```

数据查询SELECT

案例

pattern match-Regular expression：

%represents any sequence of zero or more characters (wildcard)

_represents any single character

Escape character\

```
SELECT sNo,sName,sex
FROM Student
WHERE sName LIKE ‘张%’;
```

[案例] 查询名字中第2个字为"阳"字的学生的姓名和学号。

```
SELECT Sname，Sno
 FROM    Student
 WHERE  Sname LIKE '__阳%';
```

数据查询SELECT

- 案例 pattern match-Regular expression：

[案例] 查询所有不姓刘的学生姓名、学号和性别。

```
SELECT Sname, Sno, Ssex
 FROM    Student
 WHERE  Sname NOT LIKE '刘%';
```

```
SELECT *
FROM Course
WHERE cName LIKE 'DB\_%i__' ESCAPE '\';
```

```
SELECT *
FROM Student
WHERE sName ~ '^[\u4E00-\u9FA5]{3,4}$';
```

- 正则表达式
- 中文unicode编码
- \u4E00-\u9FA5 此范围为基本汉字

数据查询SELECT

逻辑运算

```
SELECT *
FROM Student
WHERE dNo='01' AND (age<20 or age>23);
```

- 逻辑运算符：AND和 OR来连接多个查询条件
  - AND的优先级高于OR
  - 可以用括号改变优先级

[案例] 找出需要重考1号课程的学生。

```
SELECT Sno
FROM SC
WHERE Grade < 60 AND Cno='1';
```

查询结果不包括缺考的学生，因为他们的Grade值为null。

数据查询SELECT

关于空值null的查询

- 空值就是“不知道”或“不存在”或“无意义”的值。
- 一般有以下几种情况：
  - 该属性应该有一个值，但目前不知道它的具体值
  - 该属性不应该有值
  - 由于某种原因不便于填写
- “IS” 不能用 “=” 代替

```
SELECT sNo
FROM SC
WHERE cNo='1' and score is null;
```

[案例] 找出需要重考1号课程的学生。

```
SELECT Sno
FROM SC
WHERE Cno='1' AND (Grade<60 OR Grade IS NULL);
```

```
SELECT *
FROM Student
WHERE age NOT IN (18, 19, null);
```

?

数据查询SELECT

- 空值与另一个值（包括另一个空值）的算术运算的结果为空值
- 空值与另一个值（包括另一个空值）的比较运算的结果为UNKNOWN。
- 有UNKNOWN后，传统二值（TRUE，FALSE）逻辑就扩展成了三值逻辑

x	y	x AND y	x OR y	NOT x
T	T	T	T	F
T	U	U	T	F
T	F	F	T	F
U	T	U	T	U
U	U	U	U	U
U	F	F	U	U
F	T	F	T	T
F	U	F	U	T
F	F	F	F	T

T表示TRUE，F表示FALSE，

U表示UNKNOWN

数据查询SELECT

Sorting Results (ORDER BY clause)

- In general, the rows of an SQL query result table are not arranged in any particular order. We can user ORDER BY clause in the SELECT statement to sort the results.
- The ORDER BY clause must always be the last clause of the SELECT statement.
- ASC—ascending order, default order
- DESC—descending order

❖ ORDER BY子句

- 可以按一个或多个属性列排序
- 升序：**ASC**；降序：**DESC**；缺省值为升序

数据查询SELECT

- 案例

```
SELECT sNo, score
FROM SC
WHERE cNo='010101' ORDER BY score DESC;

SELECT *
FROM Student
ORDER BY dNo, age DESC;
```

Which is the place for null?

数据查询SELECT

- Using the SQL Aggregate Functions

  COUNT([DISTINCT | ALL] *)

  COUNT([DISTINCT | ALL] <columnName>)

  SUM([DISTINCT | ALL] <columnName>)
  AVG([DISTINCT | ALL] < columnName >)

  MAX([DISTINCT | ALL] < columnName >)
  MIN([DISTINCT | ALL] < columnName >)

  – COUNT, MIN, and MAX apply to both numeric and non-numeric fields, but SUM and AVG may be used on numeric field only.
  – Apart from COUNT(*), each function eliminates nulls first and operates only on the remaining non-null values.

- 将关系中的多行数据，通过聚集函数运算成一个结果展示

数据查询SELECT

- 案例

```
SELECT COUNT(*)
FROM Student;
```

```
SELECT COUNT(DISTINCT sNo)
FROM SC;
```

```
SELECT COUNT(*) AS countOf
FROM Course
WHERE credit >= 2;
```

```
SELECT SUM(credit)
FROM Course
WHERE dNo=‘001’;
```

```
SELECT AVG(score)
FROM SC
WHERE cNo='010101';
```

```
SELECT MIN(age)
FROM Student
WHERE dNo='001';
```

数据查询SELECT

Grouping Results (GROUP BY clause)

- All column names in the SELECT list must appear in the GROUP BY clause unless the name is used only in an aggregate function.
- When the WHERE clause is used with GROUP BY, the WHERE clause is applied first.
- The ISO standard considers two nulls to be equal for purposes of the GROUP BY clause.

❖ GROUP BY

- 可以按一个或多个属性列分组
- **Select**语句中能出现的列的限制

  只有被**GROUP BY** 指定的列才能出现；

  未被指定的列可以聚集函数的形式出现。

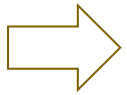
数据查询SELECT

Grouping Results (GROUP BY clause)

- All column names in the SELECT list must appear in the GROUP BY clause unless the name is used only in an aggregate function.
- When the WHERE clause is used with GROUP BY, the WHERE clause is applied first.
- The ISO standard considers two nulls to be equal for purposes of the GROUP BY clause.

```
SELECT cNo, COUNT(sNo)
FROM SC
GROUP BY cNo;
```

sNo	cNo	score
s01	001	90
s02	002	95
s01	002	80
s03	001	70
s02	003	



sNo	cNo	score	Count(sNo)
s01 s03	001	90 70	2
s01 s02	002	80 95	2
s02	003		1

数据查询SELECT

- Restricting groupings (HAVING clause)
  - HAVING clause is designed for use with the GROUP BY clause to restrict the groups that appear in the final result table.
  - The HAVING clause is not a necessary part of SQL.
  - Similar in syntax, HAVING and WHERE serve different purposes.
  - The ISO standard requires that column names used in the HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function.

❖ **HAVING** **WHERE先对成绩限制，再分组，再限定**

- **对分组GROUP BY后的数据进一步限定**
- **WHERE 对单行数据限制;HAVING 对分组后限制。**
- **HAVING 后的列名限制（GROUP BY指定的列/聚集函数 ）**

```
SELECT sNo
FROM SC
WHERE score>60
GROUP BY sNo
HAVING COUNT(*)>3;
```

数据查询SELECT

- 案例

```
SELECT dNo, COUNT(sNo)
FROM Student
GROUP BY dNo
HAVING COUNT(sNo)>100;
```

```
SELECT  MAX(score), MIN(score)
FROM SC
GROUP BY cNo;
```

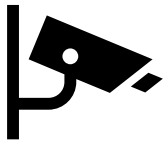
```
SELECT  AVG(age)
FROM Student
GROUP BY dNo;
```

```
SELECT  SUM(credit)
FROM Course
GROUP BY dNo;
```

数据查询SELECT

[案例]查询平均成绩大于等于90分的学生学号和平均成绩：

```
SELECT Sno, AVG(Grade)
FROM  SC
WHERE AVG(Grade)>=90
GROUP BY Sno;
```



因为WHERE子句中是不能用聚集函数作为条件表达式

正确的查询语句应该是：

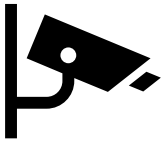
```
SELECT  Sno, AVG(Grade)
FROM  SC
GROUP BY Sno
HAVING AVG(Grade)>=90;
```

数据查询SELECT

[案例]查询各个学生的平均成绩：

```
SELECT Sno, AVG (Score)
FROM SC
GROUP BY Sno;
```

— Apart from COUNT(*), each function eliminates nulls first and operates only on the remaining non-null values.



	sno [PK] character	cno [PK] character	score integer	recorddate date
1	170101	030101	91	2016-01-08
2	170101	030102	83	2016-07-10
3	170101	020101	88	2016-07-02
4	170101	020102	92	2017-01-10
5	170101	020201	70	2017-01-10
6	170101	020202	80	2017-01-10
7	170101	020203	[null]	[null]
8	170101	020301	[null]	[null]
9	170101	020302	[null]	[null]
10	170101	020401	[null]	[null]
11	170101	020402	[null]	[null]

	sno character	avg numeric
1	170406	65.0000000000000000
2	170102	78.1111111111111111
3	170405	67.7500000000000000
4	170103	78.4285714285714286
5	170205	75.0833333333333333
6	170201	76.0000000000000000
7	170105	[null]
8	170207	[null]
9	170108	67.4285714285714286
10	170104	[null]
11	170101	84.0000000000000000
12	170202	77.2727272727272727
13	170401	80.8000000000000000

关于本讲内容

1. SQL结构化查询语言，是关系数据库的标准语言
2. 数据定义：CREATE, ALTER, DROP,
3. 数据查询：SELECT——仅涉及**一个表**：
   1. 选择表中的若干列
   2. 选择表中的若干元组
   3. ORDER BY子句
   4. Aggregate函数
   5. GROUP BY子句 Having子句

SELECT A
FROM R
WHERE F

$$\pi_A(\sigma_F(R))$$

关于本讲内容

祝各位学习愉快！

感谢观看！

讲解人：李鸿岐