SQL学习

第1部分
Part-I

投票(匿名)

最多可选1项

设置

你使用过SQL语言吗?

A 没有用过，第一次听说

B 偶尔用过

C 经常用，操作mysql数据库

D 经常用，操作其它数据库，如SQL server

提交

SQL简介

SQL introduction

SQL背景

- 关系型数据库被认为是人类社会的重大的发明，甚至被认为改变现代文明的技术之一

- SQL作为与关系型数据库标配的关系语言，一直很成功

- SQL语言从诞生至今，已经有近50年的历史，在大数据时代依然。
  - 2000年后兴起的NoSQL运动
  - 2018年后发展为NewSQL，重新认可SQL为大数据的基础

SQL诞生历史-天时地利人和

- Donald D. Chamberlin和Raymond F. Boyce于1972年加入IBM，1974年在IBM圣何塞研究院设计实现完成，最初命名为
  - SEQUEL: A STRUCIURED ENGLISH QUERY LANGUAGE
  - 简化为SQL，i.e. Structured Query Language

- 1970年，E.F.Codd提出的关系数据模型和关系代数

- 发明者之一Raymond F. Boyce英年早逝，天妒英才，令人唏嘘。

1. D. Chamberlin and R. Boyce, ‘‘**SEQUEL: A Structured English Query Language**,’’ Proc. ACM SIGFIDET Workshop on Data Description, Access, and Control, ACM Press, 1974, pp. 249–264.
2. R. Boyce and D. Chamberlin, ‘‘**Using a Structured English Query Language as a Data Definition Facility**,’’ IBM Research Report RJ1318, Dec. 1973.

SQL语言为什么能成功?

- 设计者之一Donald D. Chamberlin归结为以下4个原因:
  - 关系代数模型，提升用户与数据交互的抽象层次，提升了通用性
    - 实践证明SQL简洁，表达力强（未完全遵守著名的Codd的12原则）
  - 多平台的鲁棒实现，System R（IBM）和UNIX平台（Ingres/Oracle）
  - 单一的语言，实现查询、更新和管理任务为一体
  - SQL的标准化，提供了统一的生态，任何人都可以开发自己的实现

1. Donald D. Chamberlin, Early History of SQL, IEEE Annals of the History of Computing, 2012.

SQL语言的一些问题

- 因为成功，也带来了批评☺
- 对SQL批评主要有以下5个方面：
  - 正交性与完备性：
    - SQL 1992版本标准化完成解决了
  - 空值（Nulls）：
    - 当时被批评，后来证明有用，属于利大于弊
  - 重复数据，允许重复值出现在表中和查询结果中。
    - 回答：排序去重成本很高（当时设计时，70年代，计算机性能太弱）
  - 阻抗不匹配（Impedance Mismatch）
    - relational-object 之间的数据转换（XML，JSON）
    - 在OLTP中，SQL会嵌入到C\Python\C++中
    - 解决方法，SQL-based scripting languages
  - 过于传统（legacy）

SQL语言是什么？

- SQL是一种高级编程语言，是查询和操作数据的标准语言，语法类似英语
  - 数据定义语言 (Data Definition Language, DDL)
  - 数据操作语言 (Data Manipulation Language, DML)
  - 数据控制语言（ Data Control Language, DCL ）

- SQL是一种标准语言，标准包括:
  - ANSI SQL，SQL92 (a.k.a. SQL2)，SQL99 (a.k.a. SQL3), ….
  - SQL2003, 2006, 2008, 2011, 2016.

SQL语言是什么？

- 数据定义语言 (Data Definition Language, DDL)
  - 定义数据表的模式
  - 创建、修改、删除数据表及其属性
  - 索引，视图等等
- 数据操作语言 (Data Manipulation Language, DML)
  - 查询单表和多表
  - 插入、删除、修改数据表的元组
- 数据控制语言（Data Control Language, DCL）

SQL数据库操作工具

DBeaver：一种通用的数据库工具

关系型数据库

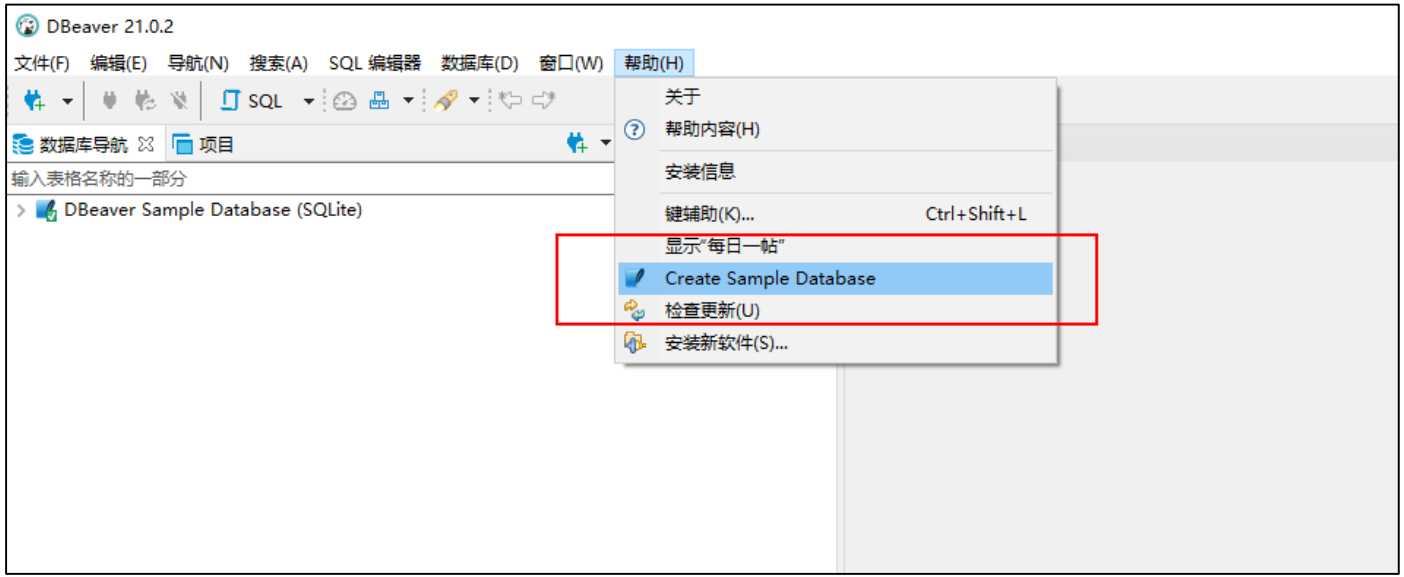
- Oracle
- Microsoft SQL Server, mSQL
- MySQL（开源），Oracle收购
- PostgreSQL（开源）
- IBM DB2
- SQLite（开源）

378 个系统排名，2021 年 9 月

秩			数据库管理系统	数据库模型	分数		
2021 年9 月	2021 年8 月	2020 年9 月			2021 年9 月	2021 年8 月	2020 年9 月
1.	1.	1.	甲骨文	关系型多模型	1271.55	+2.29	-97.82
2.	2.	2.	MySQL	关系型多模型	1212.52	-25.69	-51.72
3.	3.	3.	微软 SQL 服务器	关系型多模型	970.85	-2.50	-91.91
4.	4.	4.	PostgreSQL	关系型多模型	577.50	+0.45	+35.22
5.	5.	5.	MongoDB	文档, 多模型	496.50	-0.04	+50.02
6.	6.	↑ 7.	Redis	键值,多模型	171.94	+2.05	+20.08
7.	7.	↓ 6.	IBM Db2	关系型多模型	166.56	+1.09	+5.32
8.	8.	8.	弹性搜索	搜索引擎, 多模型	160.24	+3.16	+9.74
9.	9.	9.	SQLite	关系型	128.65	-1.16	+1.98
10.	↑ 11.	10.	卡桑德拉	宽柱	118.99	+5.33	-0.18

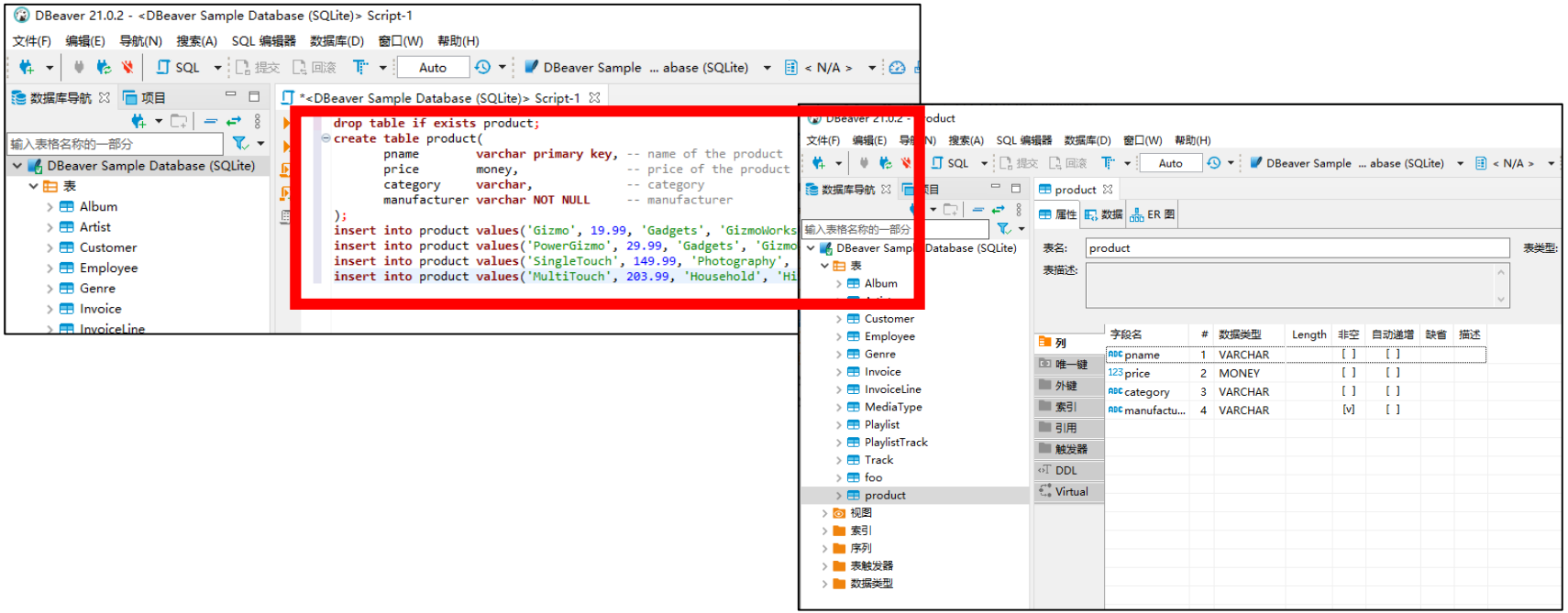
DBeaver 工具（客户端）

- DBeaver中集成的Sample Database就是使用的本地SQLite



DBeaver 操作

- 从DBeaver中可视化的查看创建的数据表:



SQL语言上手

Hands-on

SQL快速上手

- 上手学习如何快速创建数据库，创建数据表，以及最基本的增删改查处理：
- DDL语句：创建数据库，创建数据表，修改表属性
- DML语句：增加(Create)、检索(Retrieve)、更新(Update)和删除(Delete)，缩写为CRUD。

一些细节 A Few Details

- SQL 语句可以写成一行，也可以分写为多行。
  - 多条 SQL 语句必须以分号（;）分隔。
- SQL 语句时，所有空格都被忽略。
- SQL 支持三种注释
  - ## 注释
  - -- 注释
  - /* 注释 */

！！！ 注意中英文的双引号、逗号和分号

一些细节 A Few Details

- SQL 命令是大小写无关
- 相同: SELECT，Select，select
- 相同: Product， product （表名）
- **输入值是大小写相关的:**
- 属性值是大小写敏感的
- Different: ‘Seattle’，‘seattle’
- 使用单引号表示常数:

  ‘abc’ - yes

  “abc” - no

SQL快速上手

- 1.1 创建数据库（ DDL语句）
- 1.2 创建数据表（ DDL语句）
- 1.3 添加数据 （DML语句）
- 1.4 查询数据（DML语句）
- 1.5 更改数据（DML语句）
- 1.6 删除数据（DML语句）
- 1.7 连接操作（关联查询）
- 1.8 事务操作（TxB）
- 1.9 视图（View）
- 1.10 索引操作（Index）
- 1.11 键约束（Key constraints）

1.1 创建数据库

- 要创建数据库，用户必须是超级用户或具有CREATE DB特权。
- 语法：
- CREATE DATABASE创建一个新的数据库。示例：

```
CREATE DATABASE
        testdb;    --要创建的数据库名称
```

1.2 创建数据表

- CREATE TABLE在当前数据库中创建一个最初为空的数据表。
- 语法：
- CREATE TABLE 表名 (列名 列类型, …);
  - 常见列类型：
    1. bool：逻辑布尔值(true/false)
    2. int：整数类型
    3. real：浮点数类型
    4. text：字符串类型
    5. timestamp：时间类型

1.2 创建数据表

- 示例：

```
##创建表
CREATE TABLE table1                    --表名
(
        c1  INT,                --列1
        c2  INT                 --列2
);
```

```
##删除
DROP TABLE table1;
DROP TABLE if exists table1;
```

1.2 修改数据表

```
##添加列
ALTER TABLE table1
ADD c3 int(3);

##修改类型
ALTER TABLE table1
MODIFY COLUMN c3 tinyint;

##删除列
ALTER TABLE table1
DROP COLUMN c3;
```

```
## 指定主键
ALTER TABLE table1
ADD PRIMARY KEY (c1);

## 删除主键
ALTER TABLE table1
DROP PRIMARY KEY;
```

1.3 添加数据

- INSERT将新行插入数据表中，可以插入由值表达式指定的一行或多行。
-
- 语法：
- INSERT INTO 表名 [(列名, …)] VALUES (列值, …), …;

```
INSERT INTO table1
VALUES
(1, 2),
(2, 3),
(3, 4);
```

1.3 添加数据

- 示例1：添加单条数据到数据表中

```
INSERT INTO table1 VALUES(1, 2);
```

- 示例2：添加多条数据到数据表中

```
INSERT INTO table1 VALUES(1, 2),(2,3);
```

- 示例3：指定要添加的列（未指定的列去默认值）

```
INSERT INTO table1 (c1) VALUES(1);
```

1.4 查询数据

- SELECT从数据表中检索行。
-
- 语法：
- SELECT * FROM 表名 [WHERE 条件];
- SELECT 列名, … FROM 表名 [WHERE 条件];

1.4 查询数据

- 示例:

```
SELECT * FROM table1;                    --查询所有行所有列

SELECT c1, c2 FROM table1;                    --查询所有行的c1,c2列

SELECT * FROM table1 WHERE c1 = 1;      --查询满足c1=1条件的行
```

1.5 更新数据

- UPDATE用来更新数据表的行。
- 语法：
  - UPDATE 表名 SET 列 = 值, … [WHERE 条件];
  - 示例：

```
UPDATE table1 SET c1 = 1;                --更新所有行的c1列为1
UPDATE table1
        SET c1 = 1                       --要更新的列
        WHERE c2 = 2;                    --更新条件
```

1.6 删除数据

- DELETE删除数据表中的行。
- 语法：
- DELETE FROM 表名 [WHERE 条件];
- 示例：

```
DELETE FROM table1;                    --删除所有行

DELETE FROM table1
        WHERE c1 = 1;                  --删除满足条件的行
```

1.7 连接操作（JOIN）

- 关系型数据库支持连接操作（JOIN），包括：

1. 内连接：INNER JOIN
2. 左连接：LEFT JOIN
3. 右连接：RIGHT JOIN
4. 全连接：FULL JOIN

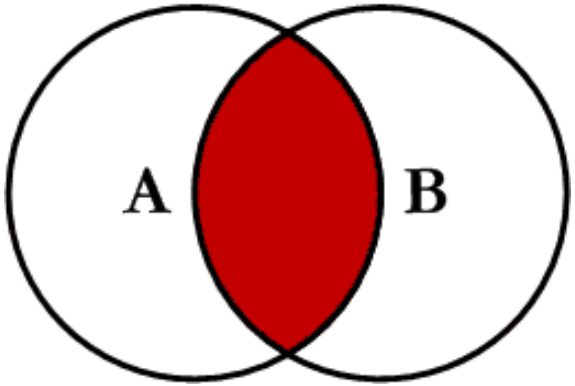
1.7 连接操作

- 假设有A和B是两张只有一列的数据表，并且有如下数据行：

```
CREATE TABLE A
(
        c1 INT
);
CREATE TABLE B
(
        c1 INT
);
INSERT INTO A VALUES(1),(2),(3);
INSERT INTO B VALUES(2),(3),(4);
```

1.7 连接操作

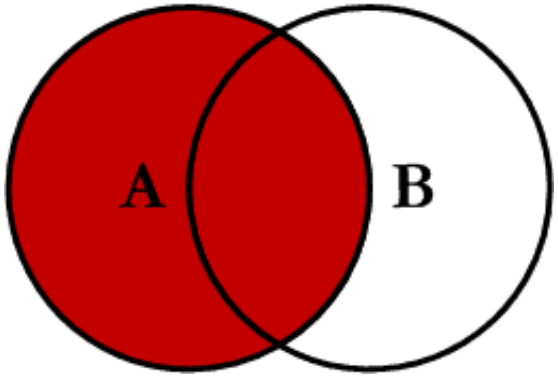
- 内连接:
  - SELECT * FROM A表 INNER JOIN B表 ON(条件)
- 内连接取满足条件的笛卡尔积




```
# select * from A inner join B on(A.c1=B.c1);
 c1 | c1
----+----
  2 |  2
  3 |  3
(2 rows)
```

1.7 连接操作

- 左连接：SELECT * FROM A表 LEFT JOIN B表 ON(条件)
- 左连接取满足条件的笛卡尔积外加表A中不满足连接条件的行，表B的列则用空值填充

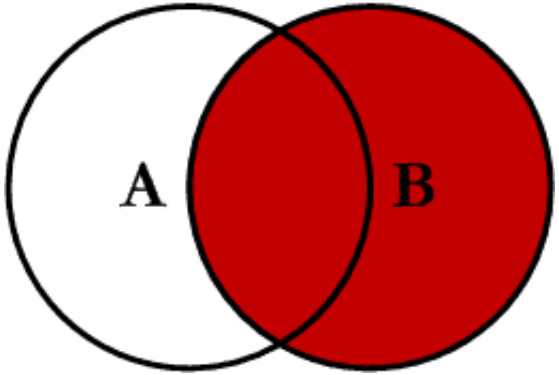



```
# select * from A left join B on(A.c1=B.c1);
 c1 | c1
----+----
  1 |
  2 |  2
  3 |  3
(3 rows)
```

1.7 连接操作

- 右连接：SELECT * FROM A表 RIGHT JOIN B表 ON(条件)

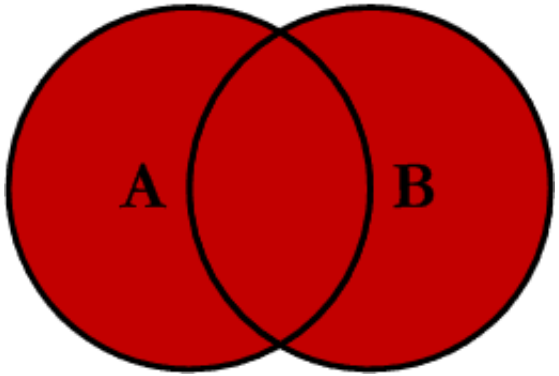
- 右连接与左连接相反，取满足条件的笛卡尔积外加表B中不满足连接条件的行，表A的列则用空值填充




```
# select * from A right join B on(A.c1=B.c1);
 c1 | c1
----+----
  2 |  2
  3 |  3
    |  4
(3 rows)
```

1.7 连接操作

- 全连接：SELECT * FROM A表 FULL JOIN B表 ON(条件)
- 全连接返回满足条件的笛卡尔积，外加表A和表B中不满足条件的行，另外一张表中的列用空值填充




```
# select * from A full join B on(A.c1=B.c1);
 c1 | c1
----+----
  1 |
  2 |  2
  3 |  3
    |  4
(4 rows)
```

想一想，练一练：

- 请在以上示例的基础上，测试一下数据表A和数据表B的inner join与left outer join执行的效果

1.8 创建索引-Index

- 索引（Index）是一种特殊的数据结构，加快数据查询（当表的行数很大时）
- 单列索引是一个基于表的一个列上创建的索引

```
CREATE INDEX index_name ON table_name (column_name);
create index idx_c on table1(c1)
```

- 组合索引是基于一个表的两个或多个列上创建的索引

```
CREATE INDEX index_name ON table_name (column1, column2);
Create index idx_cc on table1(c1,c2)
```

- 删除索引

```
DROP INDEX index_name;
```

1.9 视图-View

- 视图（VIEW）是基于 SQL 语句的结果集的可视化的表。
- 视图是虚拟的表，本身不包含数据，也就不能对其进行索引操作。
- 对视图的操作和对普通表的操作一样。

```
CREATE VIEW bigger_10_view AS
SELECT c1, c2
FROM table1
WHERE c1 > 10;
```

```
DROP VIEW bigger_10_view;
```

1.10 事务操作（TRANSACTION）

- 事务（Transaction）
  - 事务执行包含多条SQL语句，形成一个执行块
  - 执行块里的SQL语句要么全部执行成功，要么全部不执行

•BEGIN TRANSACTION：开始事务处理
•COMMIT：保存更改，同 END TRANSACTION
•ROLLBACK：回滚所做的更改

1.11 键约束（Key Constraints）

- 键是属性的最小子集可以作为关系中的元组的唯一标识符
  - 主键Primary key
  - 外键Foreign key

```
INSERT INTO A VALUES(1,4),(2,5),(3,6);
INSERT INTO B VALUES(2,4),(3,5),(4,6);
```

```
CREATE TABLE A
(
        c1 INT,
        c2 INT,
Primary KEY(c1)
);
```

```
CREATE TABLE B
(
        c1 INT,
        c2 INT,
Primary key(c1),
Foreign key(c2) references A(c2)
);
```

想一想，练一练

- 动手实践：想一想这些属性应该用什么样数据类型？
- 创建个人信息表，
  - 数据表Person，
  - 姓名name，
  - 年龄age，
  - 住址address
- 创建工作信息表，
  - 数据表Job，
  - 姓名name，
  - 工龄seniority，
  - 级别level，
  - 分公司branch

Person(name, age, address)

Job(name, seniority, level, branch)

谢谢指正！