SQL基础

@author ixenos

SQL基础·目录

1.数据定义（CREATE、DROP、ALTER）

1）**基本表**的定义（创建、修改和删除）

2）**索引**的定义（创建和删除）

3）**视图**的定义（创建和删除）

2.数据查询（SELECT）

1）**单表**查询（只涉及一个表）

投影（列）、

选择（行 ：WHERE 条件）、

聚集（列：COUNT、MAX、MIN、SUM、AVG）、

GROUP BY（HAVING）、

ORDER BY

2）**连接**查询

内连接、外连接（左外连接、右外连接）

3）**嵌套**查询（子查询）

3.数据更新（INSERT、UPDATE、DELETE）

1）**插入**数据

2）**修改**数据

3）**删除**数据

4.视图

1）**定义**视图

2）**查询**视图

3）**更新**视图

4）**撤销**视图

5.数据控制（GRANT授权、REVOKE取消授权）

1）**授予**权限

2）**回收**权限

SQL基础·SQL语句

SQL基础·详细

**0.关系模型和关系：**

关系模式是型，关系是值。

关系模式直观上就是一张空表，是对关系的描述，添上具体的值之后就是关系了

**1.数据定义（CREATE、DROP、ALTER）**

基本表、索引、视图

**0）视图是基于基本表的虚表，索引依附于基本表，通常没有支持修改这两的SQL**

**1）基本表的定义（创建、修改和删除）**

**（0）完整性约束条件**

列级完整性约束和表级完整性约束：

如果完整性约束条件涉及到多个属性列，则必须定义在表级上，否则既可以定义在列级也可以定义在表级

例如：复合主码和复合外码就只能定义在表级

PRIMARY KEY (借书证号,ISBN)

关系模型中，完整性约束含有：

实体完整性：定义表的主码

参照完整性：定义表的外码（对其他表的引用）

用户定义完整性：

数据类型、

数据格式、

取值范围、

空值约束（NOT NULL）、

缺省值（DEFAULT（X））、

唯一值（UNIQUE）

#**<>**代表必须、**[]**代表可选

**（1）创建基本表**：

CREATE TABLE <表名> (

<列名><数据类型> [列级完整性约束]

[,<列名><数据类型> []]

[,…..]

[,<表级完整性约束>]);

Example：

CREATE TABLE BORROW(

借书证号 CHAR(8) ,

ISBN CHAR(16) ,

借书时间 DATE NOT NULL ,

应还时间 DATE NOT NULL，

PRIMARY KEY (借书证号, ISBN),

FOREIGN KEY 借书证号 REFERENCES STUDENT (借书证号),

FOREIGN KEY ISBN REFERENCES BOOK (ISBN));

**（2）修改基本表**

ALTER TABLE <表名>

[ADD <新列名><数据类型>[完整性约束]]

[DROP <列名> [ RESTRICT | CASCADE ]]

[MODIFY <列名><新数据类型>] ;

注意：

安全！RESTRICT代表在没有任何视图或约束（外码约束）引用到该属性列时，才可删除

决绝！CASCADE代表在删除时，引用到该属性列的视图和约束（外码约束）也要一起删除

记住：主键唯一非空，不能改变这两个约束，而且重复定义这两约束没有意义

例子：

ALTER TABLE STUDENT **ADD** 年龄 SMALLINT NOT NULL;

ALTER TABLE STUDENT **DROP** 生日 CASCADE;

ALTER TABLE STUDENT **MODIFY** 书名 CHAR(20);

**（3）删除基本表**

DROP TABLE 表名 [ RESTRICT | CASCADE ]

例子：

DROP TABLE 临时表 CASCADE;

注意：

对于RESTRICT和CASCADE的实现，不同数据库间具有差异性（可能有可能没有）;

执行删除操作后，表中的数据就丢失了，慎用！

**2）索引的定义（创建和删除）**

**（0）建立索引能有效加快查询速度**

原理：**根据关系中的某些字段的值**建立一颗树形结构的文件。

索引文件中存储的是按照某些字段的值来排列的一组记录号，每个记录指向一个待处理的记录。

因此索引可以理解为根据某些字段的值进行逻辑排序的一组指针。

很多DBMS直接使用主键的概念建立主索引（在建表时定义主键就建立了主索引），一个表只能有一个主索引。还可建立其他索引，不同DBMS有不同的其他索引。

SQL支持用户根据应用环境的需要，在基本表中建立一个或多个索引，以提供多种存取路径，加快查找速度。

一般只有DBA和表的主人才能进行索引的创建和删除。

**（1）建立索引**

CREATE INDEX <索引名> ON <表名>

<列名> [<次序>]

[,<列名> [<次序>]]

…);

次序有ASC（升序）和DESC（降序），缺省值是ASC

例子：

CREATE INDEX STU\_IDX\_LNO ON STUDENT (借书证号 DESC);

CREATE INDEX COU\_IDX\_BNO ON BOOK (ISBN);

CREATE INDEX SC\_IDX\_LNO\_BNO ON BORROW (借书证号, ISBN ASC)

**（2）删除索引**

DROP INDEX <索引名>;

**注意**：

当一个索引被删除后，该索引先前占有的存储空间被回收，但DROP INDEX不会影响PRIMARY KEY和UNIQUE约束条件，这些约束条件必须用ALTER TABLE…DROP删除列来完成；

**DBMS一般会自动建立PRIMARY KEY和UNIQUE列上的索引**。系统在存取数据时会自动选择合适的索引作为存取路径，用户不必也不能显式地选择索引。

使用索引：提高查询速度、降低增删改的速度

**索引使用技巧**：

1.**行数越多**，索引提供的性能提升越大；反之，记录较少时，没有性能提升

2.索引列有较多**不同的数据**和**空值**时，能大大提高索引的性能

3.**查询返回的数据少**时，索引优化查询效果最好，反之数据多时，索引会加大系统开销

4.**索引使更新操作变慢**，因此**不对需要经常更新和修改的字段创建索引**，因为**更新索引**的开销很大

5.索引会**占用数据库的空间**，设计数据库可用空间时不要忘了考虑进去

6.不要将索引和表存储在同一个驱动器上，**分开存储避免访问冲突**，能使结果返回更快

**3）视图的定义（创建和删除）**

（略，请看4.）

**2.数据查询（SELECT）**

SELECT [ALL | DISTINCT] <目标列表达式> [,<目标列表达式>]…

FROM <表名或**视图名**> [,<表名或视图名>] …

[WHERE <条件表达式>]

[GROUP BY <列名1> [ HAVING <条件表达式> ]]

[ORDER BY <列名2> [ ASC | DESC ]];

#[ALL | DISTINCT] 使用DISTINCT可消除重复行，ALL是缺省值

#[ ASC | DESC ] ASC是缺省值，升序

#GROUP BY带有HAVING，则结果只有满足**指定条件**的组

**1）单表查询（只涉及一个表）**

投影（列）、

选择（行 ：WHERE 条件）、

聚集（列：COUNT、MAX、MIN、SUM、AVG）、

GROUP BY（HAVING）、

ORDER BY

**（1）选择表中的若干列（投影）**

**1] 选择指定列**

SELECT 姓名,专业名,借书数 FROM STUDENT;

**2] 全部列**

SELECT \* FROM STUDENT;

**3] 经过计算的值**

自动给计算列别名Expr：

SELECT 姓名, 2009-Year(出生时间) FROM STUDENT

我们也可用AS自定义别名：

SELECT 姓名, 2009-Year(出生时间) AS ‘年龄’ FROM STUDENT

**4] 插入字符串列**

SELECT 姓名, ‘还可借书数：’ , 6-借书数 AS ‘可借书数’ FROM STUDENT;

这样在还可借书数和可借书数两列之间插入了字符串列，而字符串列中所有值都是“还可借书数：”

**（2）选择表中的若干元组（选择）**

**1] 消除结果集中的重复行（DISTINCT）**

虽然关系数据库中不允许出现完全相同的两个元组，但是投影时得到的结果集还是可能会出现重复行！

在SELECT子句中使用DISTINCT关键字可以消除结果集中的重复行！

SELECT DISTINCT 借书证号 FROM BORROW

**2] 查询满足条件的元组（WHERE）**

下面是查询条件：

比较：=, <>, >, <, >=, <=, !=, !>, !<,或者NOT加上以上字符

算术运算： +, -, \*, /

确定范围：BETWEEN AND, NOT BETWEEN AND

确定集合：IN, NOT IN

字符匹配：LIKE, NOT LIKE

空值：IS NULL, IS NOT NULL

多重条件：AND, OR, NOT

查询条件示例：

**[1] 比较运算**

SELECT \*

FROM STUDENT

WHERE 借书数 **>=** 3;

**[2] 指定范围**

查询价格介于30到50的书籍的ISBN、书名、作者、出版社

SELECT ISBN, 书名,作者,出版社

FROM BOOK

WHERE 价格 **BETWEEN** 30 **AND** 50;

**[3] 确定集合**

查询STUDENT表中专业名为“计算机”“网络工程”“软件工程”的学生

SELECT \*

FROM STUDENT

WHERE 专业名 **IN** **(**‘计算机’**,** ’网络工程’**,** ’软件工程’**)**;

**[4] 字符匹配**

[NOT] LIKE ‘<匹配串>’ [ESCAPE ’<换码字符>’]

查找指定的属性列值与<匹配串>相匹配的行。

匹配串可以是完整字符串（LIKE相当于=），也可以是通配符（此时是**模糊查询**）

通配符有：% 和 \_

%代表0个或多个字符，\_代表1个字符，

ESCAPE代表转义符，即 \ ，用于对字符串本来就有的%和\_进行转义

示例：

查询STUDENT表中电子专业的学生的借书证号、姓名和借书数

SELECT 借书证号,姓名,借书数

FROM STUDENT

WHERE 专业名 LIKE ‘电子’;

**注意：**这里LIKE相当于=，因此当匹配串中不含通配符时，LIKE换成=，NOT LIKE换成<>或!=

查询STUDENT表中所有**姓王的**学生的借书证号、姓名、专业名和性别

SELECT 借书证号,姓名,专业名,性别

FROM STUDENT

WHERE 姓名 LIKE ‘王%’;

查询STUDENT表中名字**第二字**为“尔”的学生的详细情况

SELECT \*

FROM STUDENT

WHERE 姓名 LIKE ‘\_尔%’;

注意：如果要查询的匹配串本身就有%和\_，那么就要使用ESCAPE进行转义

查询BOOK表中书名以**100%**结束的书籍的ISBN和出版社

SELECT ISBN,出版社

FROM BOOK

WHERE 书名 LIKE ‘100%’ ESCAPE ‘%’;

**[5] 空值**：判断空值用IS NULL，不使用 =

SELECT \*

FROM STUDENT

WHERE 专业名 IS NULL;

**[6] 多重条件查询**：使用AND或OR来连接条件，**前者优先级高于后者**，可用括号改变优先级

查询BOOK表中价格在40元以上的电子出版社出版的书 **和** 所有清华出版社的书的ISBN、书名、出版社和价格

SELECT ISBN, 书名, 出版社, 价格

FROM BOOK

WHERE 出版社=’电子出版社’ AND 价格>40 **OR** 出版社=’清华出版社’;

根据优先级，OR前面的先判断，得到结果再与OR后面的执行“或逻辑”，而这也就是SQL注入的漏洞之一，这里整个WHERE条件总体成了OR条件

**（3）使用聚集函数（在投影区）**

1] COUNT：统计列值的个数 或 元组的个数

COUNT([DISTINCT|ALL] <列名>) 统计列值的个数

COUNT([DISTINCT|ALL] \*) 统计元组的个数

2] SUM和AVG： 求列值的总和 或 平均值

SUM([DISTINCT|ALL] <列名>)

AVG([DISTINCT|ALL] <列名>)

3] MAX和MIN：求列值的最大值 或 最小值

MAX([DISTINCT|ALL] <列名>)

MIN([DISTINCT|ALL] <列名>)

示例：

查询读者总人数，结果集以列名为“总人数”的列存储

SELECT COUNT(\*) AS ‘总人数’

FROM STUDENT;

查询借阅了图书的学生数，用DISTINCT消除了借书证号的冗余

SELECT COUNT(DISTINCT 借书证号) AS ‘借阅了图书的学生数’

FROM BORROW;

查询图书总册数和库存图书总册数

SELECT SUM(复本量) AS ‘图书总册数’ , SUM(库存量) AS ‘库存图书总册数’

FROM BOOK;

查询

SELECT MAX(借书数) AS ‘借书最多册数’，MIN(借书数) AS ‘借书最少册数’ FROM BOOK

WHERE 专业名=’计算机’ OR 专业名=’电子’;

（4）GROUP BY子句

GROUP BY子句用于**对查询结果**的各行按某一列或多列的值进行**分组**，值相等的为一组。

分组后还要筛选，用HAVING短语

示例：

查询STUDENT表中**各个专业**的学生数

SELECT 专业名, COUNT(借书证号) AS ‘学生数’

FROM STUDENT

GROUP BY 专业名;

这里以同专业名的为一组来分组，并以对借书证号的统计作为学生数的统计，当然这里也可以COUNT(\*)，因为是在STUDENT表中

**GROUP BY子句 搭配 HAVING短语**

示例：从学生表中找出那些女生数少于等于5个的专业，并分组

SELECT 专业名, count(\*) AS ‘女生人数’

FROM STUDENT

WHERE 性别=’女’

GROUP BY 专业名 HAVING COUNT(\*) <= 5;

这里，WHERE子句与GROUP BY…HAVING的作用对象不同，前者作用于基本表或视图，筛选FROM指定的数据对象，后者作用于WHERE限制的结果，进行分组，HAVING对分组数据进行过滤

（5）ORDER BY子句

ASC升序（默认），DESC降序(注意：空值元组最先显示)

示例：结果集按价格降序

SELECT \*

FROM BOOK

WHERE 出版社=’水利水电出版社’

ORDER BY 价格 DESC;

示例：

SELECT \*

FROM STUDENT

ORDER BY 专业名, 出生年份 DESC;

**2）连接查询**

（1）等值连接和非等值连接查询

自然连接是等值连接的特例（属性列相同的等值）

WHERE子句中用来**连接两个表的条件**（WHERE子句内容）称为**连接条件**，格式如下：

1、[<表名1>.]<列名1><比较运算符>[<表名2>.]<列名2>

2、[<表名1>.]<列名1>BETWEEN[<表名2>.]<列名2>AND[<表名2>.]<列名3>

注意：连接的各字段类型必须是可比的，但字段名不必相同

示例：查询每个读者的基本信息和借书情况

SELECT STUDENT.\*, BORROW.\*

FROM STUDENT, BORROW

WHERE STUDENT.借书证号 = BORROW.借书证号;

示例：自然连接（上面的实际上已经是自然连接）

SELECT STUDENT.借书证号, 姓名, 专业名, 性别, 借书数, 出生年份, ISBN,结束时间, 应还时间

FROM STUDENT, BORROW

WHERE STUDENT.借书证号 = BORROW.借书证号;

（2）自身连接查询

如果要在一个表中查找具有相同列值的行，可使用自身连接，使用自身连接，需要为表指定两个别名，且对所有的引用都要用到别名限定

示例：

SELECT B1.借书证号, B1.ISBN, B2.ISBN, B2.借书时间

FROM BORROW B1, BORROW B2

WHERE B1.借书证号 = B2.借书证号

AND

B1.结束时间 = B2.借书时间

AND

B1.ISBN <> B2.ISBN;

（3）外连接查询

左外连接列出连接语句左边关系中所有的元组，如果连接语句右边关系中没有与之匹配的元组，则在对应右边关系的属性字段填NULL，右外连接反之。

示例：

SELECT STUDENT.借书证号, 姓名, 专业名, 性别, 借书数, 出生年份, ISBN, 借书时间, 应还时间

FROM STUDENT

LEFT OUT JOIN

BORROW

ON (STUDENT.借书证号=BORROW.借书证号);

（4）复合条件连接查询

WHERE子句中有多个条件

示例：

SELECT STUDENT.借书证号, 姓名, 专业名, BORROW.ISBN, 书名

FROM STUDENT, BORROW, BOOK

WHERE STUDENT.借书证号=BORROW.借书证号 AND BOOK.ISBN = BORROW.ISBN AND 专业名 = ‘计算机’;

（5）集合运算查询

简单查询的结果就是元组的集合，多个结果集就可以进行集合的操作

集合操作：UNION并、INTERSECT交、EXCEPT补

示例：

(SELECT 借书证号, ISBN, 借宿时间, 应还时间

FROM BORROW

WHERE ISBN=’123123124’)

**UNION**

(SELECT 借书证号, ISBN, 借宿时间, 应还时间

FROM BORROW

WHERE ISBN=’12yrtwer24’);

3）嵌套查询

3.数据更新（INSERT、UPDATE、DELETE）

1）**插入**数据

2）**修改**数据

3）**删除**数据

4.视图

1）**定义**视图

2）**查询**视图

3）**更新**视图

4）**撤销**视图

5.数据控制（GRANT授权、REVOKE取消授权）

1）**授予**权限

2）**回收**权限