第五讲 SQL高级查询技术

教学内容

- **5.1 Where关联表**
  - 关联表查询
  - 关联表统计
- **5.2 Union查询**
- **5.3 子查询**
  - **In、Some/Any、ALL、Exists**
- **5.4 连接查询**
  - 内联接、外联接、全联接
  - 自联接
- **5.5 经典SQL语句**
  - 批量插入、关联表更新

5.1.1 双表关联查询

- 查询所有班级和学生信息

```
Select *
From ClassInfo,StudInfo
Where ClassInfo.ClassID=StudInfo.ClassID;
```

- 查询两个表中关心的字段信息

```
Select StudNo,StudName,StudGender,StudBirthDay,
       ClassInfo.ClassID ,ClassName
From ClassInfo,StudInfo
Where ClassInfo.ClassID=StudInfo.ClassID;
```

5.1.1 双表关联查询

- 使用表别名进行双表关联查询

```
Select StudNo,StudName,StudGender,StudBirthDay,
       C.ClassID ,ClassName
From ClassInfo AS C,StudInfo S
Where C.ClassID=S.ClassID ;
```

- 双表关联且条件筛选查询

```
Select StudNo,StudName,StudGender,StudBirthDay,
      C.ClassID,ClassName
From ClassInfo AS C,StudInfo S
Where C.ClassID=S.ClassID And StudName Like '李%' ;
```

5.1.2 多表关联查询

- 查询学生基本信息、班级信息和成绩信息

```
SELECT StudInfo.StudNo,StudName,
                    StudGender,StudBirthDay,
             ClassInfo.ClassID,ClassName,
             CourseID,StudScore
FROM ClassInfo,StudInfo,StudScoreInfo
WHERE ClassInfo.ClassID=StudInfo.ClassID And
             StudInfo.StudNo=StudScoreInfo.StudNo ;
```

多表关联查询—使用表别名

- 查询学生信息、班级信息、课程信息和成绩信息

```
SELECT S.StudNo,StudName,StudGender,StudBirthDay,
       C.ClassID,ClassName,
       CI.CourseID,CourseName,CourseType,CourseCredit,
       StudScore
FROM ClassInfo C,StudInfo S,
     CourseInfo CI,StudScoreInfo SI
WHERE C.ClassID=S.ClassID
      And S.StudNo=SI.StudNo
      And CI.CourseID=SI.CourseID ;
```

5.1.3 关联表统计

- 统计各学生平均分，包含学号、姓名、平均分信息

```
SELECT S.StudNo,StudName,
            AVG(StudScore) AS AvgScore
FROM StudInfo S,StudScoreInfo SI
WHERE S.StudNo=SI.StudNo
GROUP BY S.StudNo,StudName;
```

关联表统计

- 统计各学生平均分，包含学生信息和班级信息

```
SELECT S.StudNo,StudName,StudGender,ClassName,
       Max(StudScore) MaxScore,
       Min(StudScore) MinScore,
       Count(*) CourseCount,
      AVG(StudScore) AS AvgScore
FROM ClassInfo C,StudInfo S,StudScoreInfo SI
WHERE C.ClassID=S.ClassID
          And S.StudNo=SI.StudNo
GROUP BY S.StudNo,StudName,StudGender,ClassName;
```

5.2 使用UNION子句

- 语法

```
SELECT 语句
UNION [ALL]
SELECT 语句
```

- 功能
  - UNION联接多个结果集
- 使用UNION条件
  - 具有相同的结构
  - 字段数目相同
  - 结果集中相应字段的数据类型必须兼容

Union联接多个结果集

- 查询[60,70]和[90,100]之间的学生成绩

```
Select StudNo,CourseID,StudScore
From StudScoreInfo
Where StudScore>=60 And StudScore<=70
Union All
Select *
From StudScoreInfo
Where StudScore>=90 And StudScore<=100 ;
```

Union联接不同的结果集

- 使用Union联接学生、课程、班级信息

```
Select StudNo,StudName
From StudInfo
Where StudGender='男'
Union
Select CourseID,CourseName
From CourseInfo
Union
Select ClassName,'班级'
From ClassInfo ;
```

Union联接多个结果集

- 统计学号为20191152010各分数段课程门数

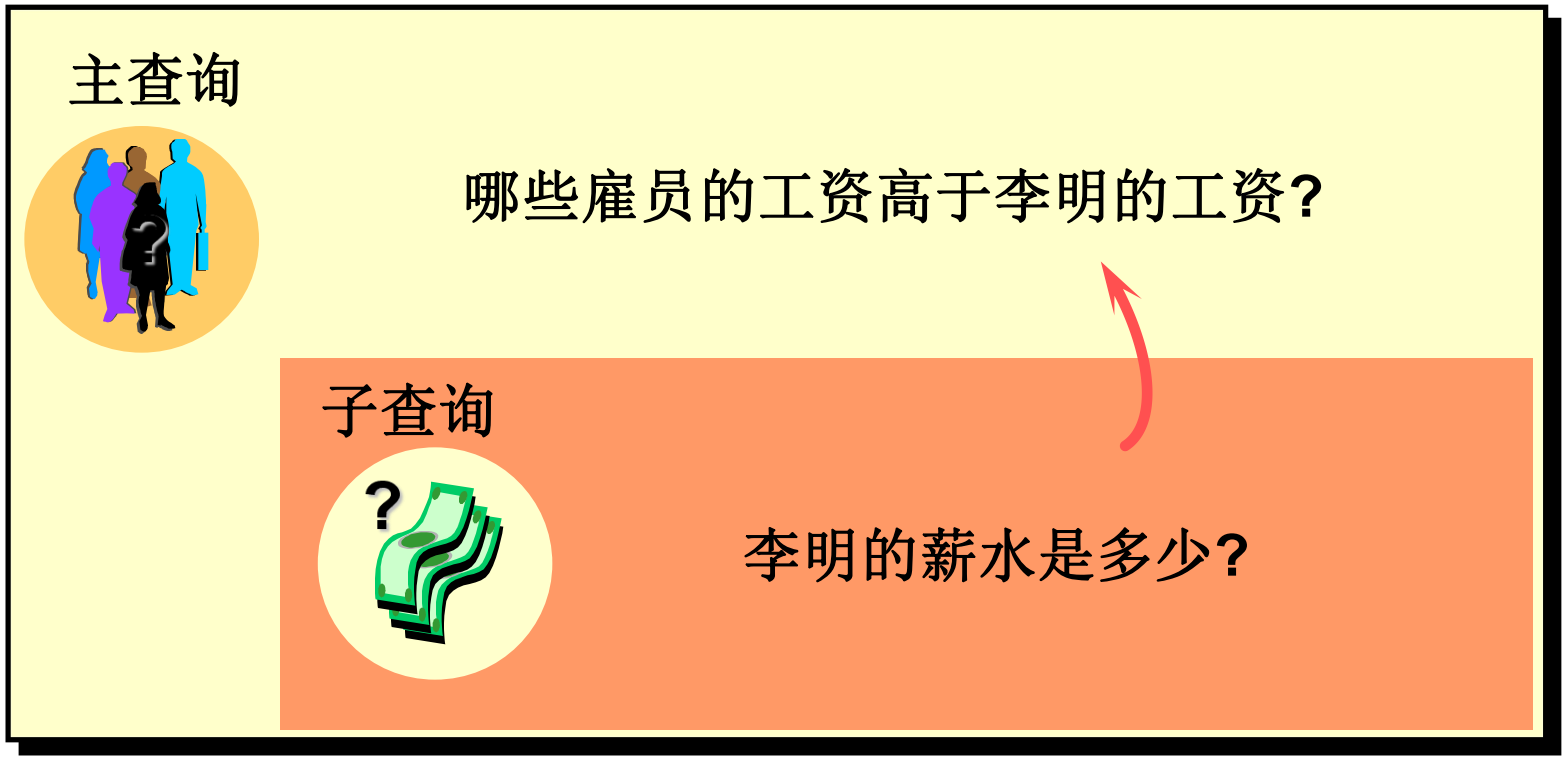
```
Select '优秀' AS 等级,'[90,100]' AS 分数段,Count(*) AS 门数
From StudScoreInfo
Where StudNo='20191152010'
        And StudScore Between 90 And 100
Union
Select '良好','[70,90)',Count(*)
From StudScoreInfo
Where StudNo='20191152010'
        And StudScore>=70 And StudScore<90 ;
```

5.3 子查询

- 嵌套查询是指在一个外层查询中包含有另一个内层查询，其中，外层查询称为主查询，内层查询称为子查询。
- 子查询作为主查询的数据来源或搜索条件。
- 子查询通常包括以下组件：
  - 标准**SELECT**查询
  - 标准**FROM**子句
  - 可选的**WHERE**子句
  - 可选的**GROUP BY**子句
  - 可选的**HAVING**子句

用子查询解决问题

- 谁的薪水比李明的多?



子查询语法

- 子查询(内查询) 在主查询之前执行一次
- 子查询的结果被用于主查询(外查询)

```
SELECT   select_list
FROM    table
WHERE expr  operator
                    (SELECT select_list
                     FROM  table) ;
```

子查询说明

- 子查询可应用于
  - **WHERE** 子句
  - **HAVING** 子句
  - **FROM** 子句
  - **CREATE VIEW** 语句
  - **UPDATE** 语句
  - **INSERT INTO** 子句
- **operator** 包括比较条件
  - 单行运算符 **(>, =, >=, <, <>, <=)**
  - 多行运算符 **(IN, SOME/ANY, ALL,EXISTS)**

使用子查询的原则

- 子查询放在圆括号中
- 将子查询放在比较条件的右边
- 在子查询中不包含**ORDER BY**子句，除非使用**SET TOP**子句
- 在单行子查询中用单行运算符
- 在多行子查询中用多行运算符

子查询的类型

- 单行子查询

- 多行子查询

单行子查询

- 仅返回一行
- 使用单行比较符

运算符	含义
=	等于
>	大于
>=	大于等于
<	小于
<=	小于等于
<>	不等于

执行单行子查询

- 查询学号为20191152010的课程成绩大于学号为20191152011课程为JSJYL011的成绩信息

```
Select * From StudScoreInfo
Where StudNo='20191152010'
        And StudScore> (
                Select StudScore
                From StudScoreInfo
                Where StudNo='20191152011'
                        And CourseID='JSJYL011');
```

在子查询中使用聚合函数

- 查询学号为20191152010的课程成绩等于学号为20191152011最高分的成绩信息

```
Select * From StudScoreInfo
Where StudNo='20191152010'
        And StudScore=(
                Select MAX(StudScore)
                From StudScoreInfo
                Where StudNo='20191152011') ;
```

多行子查询

- 返回多于一行
- 使用多行比较符

操作	含义
IN	等于列表中的任何成员
SOME/ANY	比较子查询返回的每个值
ALL	比较子查询返回的全部值
EXISTS	判断结果集行是否存在

SOME/ANY关键字

- 语法

```
expression
{ = | < > | != | > | > = | ! > | < | < = | ! < }
{ SOME | ANY } ( subquery )
```

- 功能
  - 将一个表达式的值或列值与子查询返回的一列值中的每一个进行比较
  - 如果哪行的比较结果为真，就返回该行

Some/Any示例

- 查询学生成绩高于课程最低分的成绩信息

```
Select * From StudScoreInfo
Where StudScore>any(Select StudScore
                    From StudScoreInfo);
```

- 使用单值比较运算符，执行结果相同

```
Select * From StudScoreInfo
Where StudScore>(Select min(StudScore)
                 From StudScoreInfo);
```

ALL关键字

- 语法

```
expression
{ = | <> | != | > | >= | !>| < | <= | !< }
ALL ( subquery )
```

- 功能
  - 把列值与子查询结果进行比较
  - 所有的列的查询结果都为真时返回对应行

ALL示例

- 查询所有学生成绩最高的成绩信息

```
Select * From StudScoreInfo
Where StudScore>=All(Select StudScore
                     From StudScoreInfo);
```

- 使用单值比较运算符，执行结果相同

```
Select * From StudScoreInfo
Where StudScore>=(Select max(StudScore)
                  From StudScoreInfo);
```

IN关键字

- 语法

```
test_expression [ NOT ] IN
(
   Subquery | expression [ ,...n ]
)
```

- 功能
  - IN关键字通常应用于子查询
  - 一般使用SELECT语句选定一个范围，将选定的范围作为IN关键字的符合条件的列表，从而得到最终的结果集。

IN示例

- 查询姓曹的成绩信息

```
Select StudNo
From StudInfo
Where StudName like '曹%' ;
```

```
20190505027
20190704037
20190708040
```

```
Select *
From StudScoreInfo
Where StudNo IN(
        Select StudNo
        From StudInfo
        Where StudName like '曹%';
)
```

在多行子查询中使用IN运算符

- 使用in列表查询

```
Select * From StudInfo
Where StudName  in
        ('董云霞','董文君','董晓燕','董丽华') ;
```

- 使用in子查询

```
Select *
From StudInfo
Where StudName in (
        Select StudName
        From studinfo
        Where studname like '董%');
```

使用IN子查询

- 使用in子查询学生平均分大于80的学生信息

```
Select *
From StudInfo
Where StudNo IN(
    Select StudNo
    From StudScoreInfo
    Group by StudNo
    Having Avg(StudScore)>80 );
```

使用IN子查询

- 查询重修15门以上的学生信息

```
Select *
From StudInfo
Where StudNo in(
        Select StudNo
        From StudScoreInfo
        Where StudScore<60
        Group By studno
        Having Count(*)>=15
    );
```

IN多字段子查询

- 查询同名同性别学生信息

```
Select *
From studinfo
Where concat(StudName,StudGender)  IN (
        Select concat(StudName,StudGender)
        From StudInfo
        Group By Studname,StudGender
        Having Count(*)>1
 );
```

IN多重子查询

- 查询同班同姓名的学生成绩信息

```
Select * From StudScoreInfo
 Where StudNo IN(
    Select StudNo
    From StudInfo
    Where Concat(StudName,ClassID) IN(
        Select Concat(StudName,ClassID)
        From StudInfo
        Group By Concat(StudName,ClassID)
        Having Count(*)>1));
```

EXISTS关键字

- 语法
  - EXISTS subquery
- 功能
  - 指定一个子查询，检测行的存在
  - exists搜索条件不真正使用子查询的结果
  - exists子查询中的select子句可用任意列名或用*号

```
Select * From StudInfo
Where Exists(Select *
             From StudScoreInfo
             Where StudScoreInfo.StudNo=StudInfo.StudNo
                   And StudScore=100);
```

5.4 连接查询

- 语法

```
FROM join_table join_type join_table
[ON (join_condition)]
```

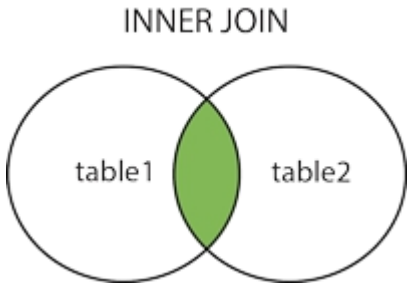
- 参数
  - **join_table**：连接的表名，可以是同一个表或多表
  - **join_type**：连接类型，内连接、外连接和交叉连接
  - **ON (join_condition)**：连接条件，由连接表中的列和比较运算符、逻辑运算符等构成。
- 注意
  - 连接不支持text、ntext和image数据类型列

内联接

- 语法

```
Select Select_list
From {< table_source >< join_type > < table_source > [ ,...n ]
      ON < search_condition >}
< join_type > ::= INNER JOIN
```

- 功能
  - 只有满足联接条件的元组才能出现在结果关系中



内联接示例

- 查询学生基本信息和成绩信息

```
Select *
From StudInfo Inner Join StudScoreInfo
     On StudInfo.StudNo=StudScoreInfo.StudNo;
```

```
Select *
From StudInfo,StudScoreInfo
Where StudInfo.StudNo=StudScoreInfo.StudNo;
```

内联接示例

- 查询两个表中的关心的字段信息

```
Select StudInfo.StudNo, StudName, StudGender,
       CourseID, StudScore
From StudInfo Inner Join StudScoreInfo
      On StudInfo.StudNo=StudScoreInfo.StudNo;
```

- 使用表别名查询两个表中的关心的字段信息

```
Select S.StudNo , StudName, StudGender,
       CourseID, StudScore
From StudInfo S Inner Join StudScoreInfo SI
       On S.StudNo=SI.StudNo;
```

内联接示例

- 多表内联接且条件筛选

```
SELECT StudInfo.StudNo,StudName,
                    StudGender,StudBirthDay,
            ClassInfo.ClassID,ClassName,
            CourseID,StudScore
FROM ClassInfo Inner Join StudInfo
                On ClassInfo.ClassID=StudInfo.ClassID
           Inner Join StudScoreInfo
                On StudInfo.StudNo=StudScoreInfo.StudNo
Where StudName Like '李%';
```

外联接

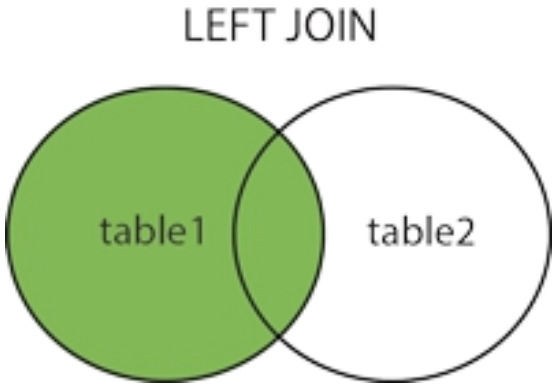
- 语法

```
Select Select_list
From {< table_source >< join_type > < table_source > [ ,...n ]
      ON < search_condition > }
< join_type > ::= LEFT | RIGHT  JOIN
```

- 功能
  - 外连接不只列出与连接条件相匹配的行
  - 列出左表(左外连接时)、右表(右外连接时)或两个表(全外连接时)中所有符合搜索条件的数据行。

左联接

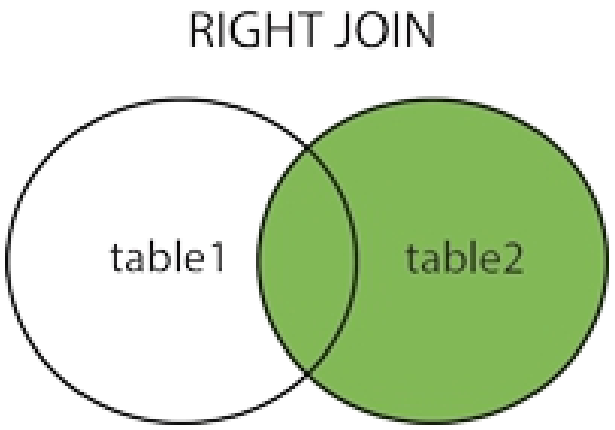
- 左联接
  - 使用LEFT JOIN或LEFT JOIN子句
  - 左向外联接，返回左表的所有行
  - 右表不匹配行返回空值
- 示例
  - 查询所有的学生信息和学生成绩信息




```
Select StudInfo.StudNo,StudInfo.StudName,
       StudScoreInfo.CourseID,StudScoreInfo.StudScore
From StudInfo Left  Join StudScoreInfo
      On StudInfo.StudNo=StudScoreInfo.StudNo
#WHERE StudScoreInfo.StudNo IS NULL #左表独有
```

右联接

- 右联接
  - 使用RIGHT JOIN或RIGHT JOIN子句
  - 左向外联接的反向联接，返回右表的所有行
  - 左表不匹配行返回空值
- 示例
  - 查询学生信息和所有的学生成绩信息




```
Select StudInfo.StudNo,StudInfo.StudName,
       StudScoreInfo.CourseID,StudScoreInfo.StudScore
From StudInfo RIGHT  Join StudScoreInfo
       On StudInfo.StudNo=StudScoreInfo.StudNo
#WHERE StudInfo.StudNo IS NULL #右表独有
```

全联接

- 全联接
  - 使用UNION子句
  - 返回左表和右表中的所有行
  - 不匹配的行返回空值
- 示例
  - 查询所有的学生信息和所有的学生成绩信息

```
Select StudInfo.StudNo,StudInfo.StudName,StudScoreInfo.CourseID,StudScoreInfo.StudScore
From StudInfo LEFT Join StudScoreInfo
        On StudInfo.StudNo=StudScoreInfo.StudNo
UNION
Select StudInfo.StudNo,StudInfo.StudName, StudScoreInfo.CourseID,StudScoreInfo.StudScore
From StudInfo RIGHT Join StudScoreInfo
        On StudInfo.StudNo=StudScoreInfo.StudNo;
```

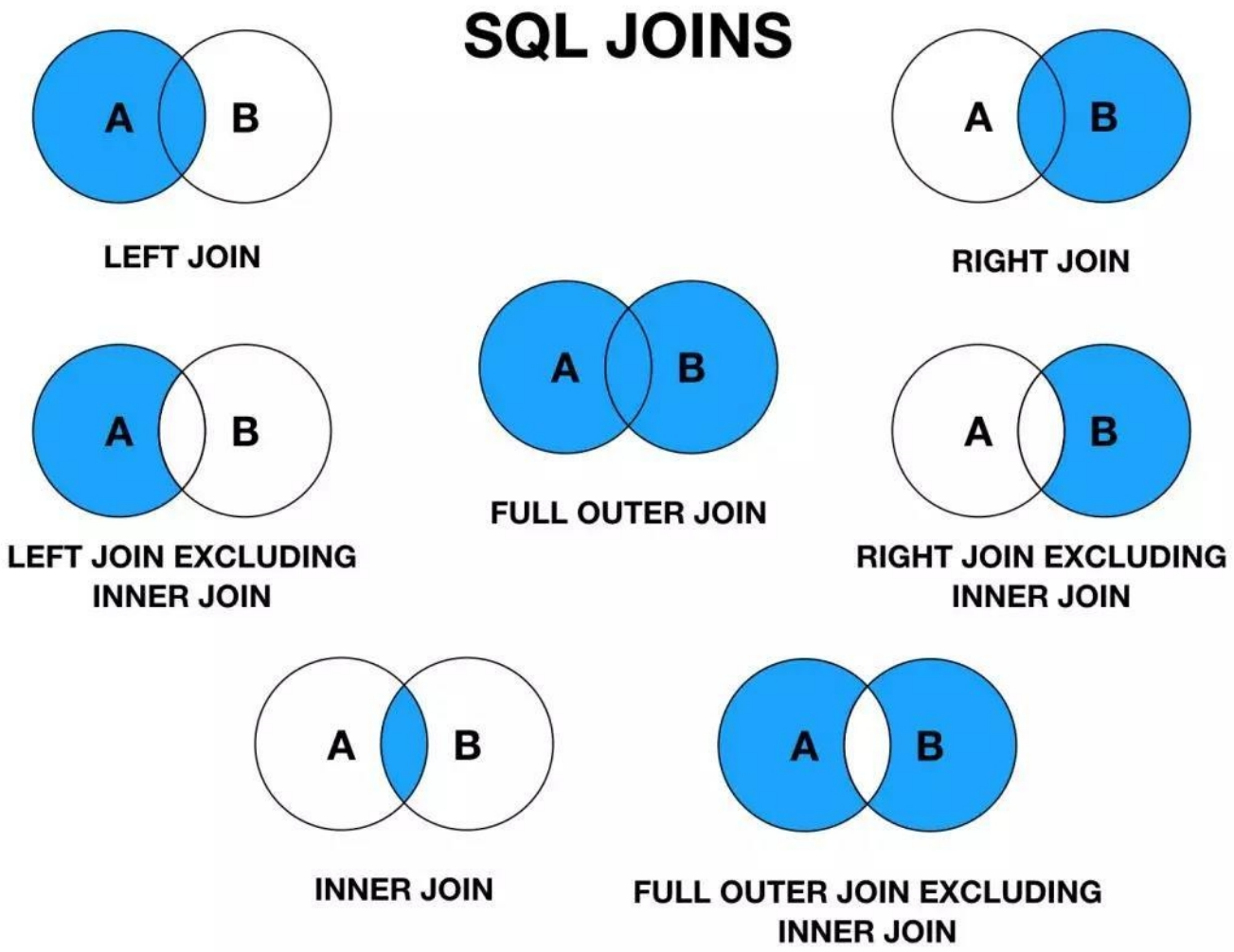
交叉连接

- 交叉连接
  - 返回连接表中所有数据行的笛卡尔积
  - 两个关系中所有元组的任意组合（M*N）
- 示例
  - 使用交叉连接查询学生信息和班级信息

```
Select *
From StudInfo Cross Join ClassInfo;
```

```
Select *
From StudInfo,ClassInfo;
```

SQL连接



自连接

- 连接不仅可以在表之间进行也可以使一个表同其自身进行连接，这种连接称为自连接（**Self Join**），相应的查询称为自连接查询。
- 注意
  - 自连接必须使用表别名运算
  - 一般用于反映上下级关系

自连接示例

行政区划表(Dic_Area)

字段名称	数据类型	字段长度	PK	字段描述	举例
Area_ID	Varchar	20	Y	行政区编码	5301
Area_Name	Varchar	30		行政区名称	官渡区
Parent_ID	Varchar	20		上级行政区编码	53

行政区划表(Dic_Area)数据

Area_ID	Area_Name	Parent_ID
53	云南省	0
5301	昆明市	53
530101	官渡区	5301
530102	盘龙区	5301
5302	安宁市	53

自连接示例

- CREATE TABLE Dic_Area(
- Area_ID varchar(20) primary key,
- Area_Name varchar(30) not null,
- Parent_ID varchar(20) not null);
- insert into Dic_Area values ('53','云南省','0');
- insert into Dic_Area values ('5301','昆明市','53');
- insert into Dic_Area values ('530101','盘龙区','5301');
- insert into Dic_Area values ('530102','五华区','5301');
- insert into Dic_Area values ('51','四川省','0');
- insert into Dic_Area values ('5101','成都市','51');

自连接示例

- 使用自连接查询行政区划的上下级关系

```
Select p.Area_ID 上级编号,p.Area_Name 上级名称,
       d.Area_ID 下级编号,d.Area_Name 下级名称
From Dic_Area p,Dic_Area d
Where d.parent_id=p.area_id;
```

行政区划(Dic_Area)自连接查询

上级编号	上级名称	下级编号	下级名称
53	云南省	5301	昆明市
5301	昆明市	530101	官渡区
5301	昆明市	530102	盘龙区
53	云南省	5302	安宁市

MySQL生成表结构语句

- Mysql>SHOW CREATE TABLE StudInfo \G;

```
mysql> SHOW CREATE TABLE StudInfo \G;
*************************** 1. row ***************************
       Table: StudInfo
Create Table: CREATE TABLE `studinfo` (
  `StudNo` varchar(15) NOT NULL COMMENT '学号',
  `StudName` varchar(20) NOT NULL COMMENT '姓名',
  `StudGender` char(2) NOT NULL DEFAULT '男' COMMENT '性别',
  `StudBirthDay` date DEFAULT NULL COMMENT '生日',
  `ClassID` varchar(10) NOT NULL COMMENT '班级编号',
  PRIMARY KEY (`StudNo`),
  KEY `FK_CID` (`ClassID`),
  CONSTRAINT `FK_CID` FOREIGN KEY (`ClassID`) REFERENCES `classinfo` (`ClassID`)
,
  CONSTRAINT `studinfo_chk_1` CHECK ((`StudGender` in (_utf8mb4'男',_utf8mb4'女'
)))
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

复制表结构+数据

- 复制结构
  - CREATE TABLE NewTableName LIKE SourceTableName
  - CREATE TABLE newTable [AS] SELECT ... FROM Table... LIMIT 0
- 复制结构和数据
  - CREATE TABLE newTable [AS] SELECT ... FROM Table...
  - LIMIT offset, count
- 示例
  - CREATE TABLE StudInfoBK LIKE StudInfo;
  - CREATE TABLE StudInfoBack AS

    Select StudNo,StudName as 姓名,StudGender,ClassID

    From StudInfo Limit 5;

复制数据表+自动ID

- 自动序列
  - CREATE TABLE StudBack
  - (AID int not null auto_increment primary key) AS
  - Select StudNo,StudName as 姓名,StudGender,ClassID
  - From StudInfo Limit 5;
- 重置序列
  - ALTER TABLE StudBack DROP AID;
  - ALTER TABLE StudBack ADD AID INT UNSIGNED NOT NULL AUTO_INCREMENT FIRST, ADD PRIMARY KEY (AID);
- 设置序列初始值
  - ALTER TABLE StudBack AUTO_INCREMENT = 100;

从另一个表中复制行—批量插入记录

- 语法
  - **INSERT INTO *table* [*column* (, *column*)] *subquery;***
- 注意
  - 不用**VALUES**子句
  - 在子查询中列数目要匹配**INSERT**子句中列的数目
- 示例

```
Insert Into StudInfoBack
        Select  StudNo,concat('姓名',StudName),
                        StudGender,ClassID
        From StudInfo
        Order By StudBirthDay Desc
        LIMIT 5;
```

关联表更新记录

- 使用学生信息表(StudInfo)中的姓名更新StudInfoBack信息表姓名字段

```
Update StudInfoBack join StudInfo
on StudInfo.StudNo=StudInfoBack.StudNo
Set 姓名=StudInfo.StudName;
```

关联表统计—使用计算字段

```
Select S.StudNo,S.StudName,
       Cast(Avg(StudScore) As Decimal(5,1)) AvgScore,
       Sum(StudScore) As SumScore,
       Max(StudScore) As MaxScore,
       Min(StudScore) As MinScore,
      Count(*) CourseCount,
       (Sum(StudScore)-Max(StudScore)-Min(StudScore))/
            (Count(*)-2) AS AvgScore2
From StudScoreInfo SS,StudInfo S
Where SS.StudNo=S.StudNo
Group By S.StudNo,S.StudName;
```

关联表统计—使用结果集

```
Select * From StudInfo S,(Select StudNo,
        Cast(Avg(StudScore) As Decimal(5,1)) AvgScore,
        Sum(StudScore) As SumScore,
        Max(StudScore) As MaxScore,
        Min(StudScore) As MinScore,
        Count(*) CourseCount,
        (Sum(StudScore)-Max(StudScore)-Min(StudScore))/
            (Count(*)-2) AS AvgScore2
    From StudScoreInfo
    Group By StudNo)  AS StudTotalScore
Where S.StudNo= StudTotalScore.StudNo
```

下次课内容

- 视图介绍
  - 视图概念
  - 视图优点
  - 视图注意事项
- 创建视图
  - 使用SQL创建视图
  - 使用Navicat创建视图
- 使用视图
  - 查询视图数据
  - 视图关联表查询
- 修改和删除视图