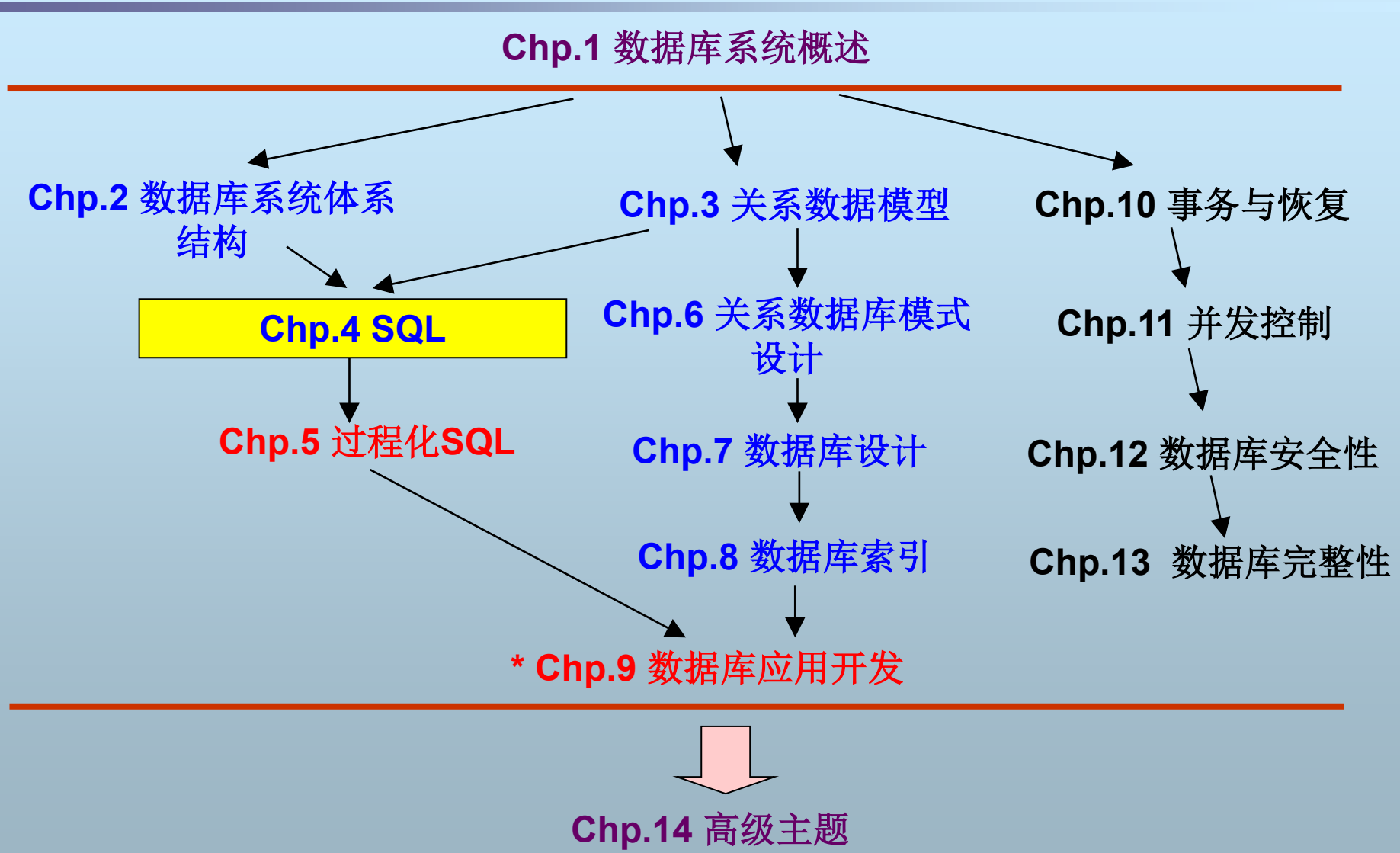
第4章 关系数据库语言SQL

金培权（jpq@ustc.edu.cn）

课程知识结构


Chp.1 数据库系统概述
Chp.2 数据库系统体系结构
Chp.3 关系数据模型
Chp.10 事务与恢复
Chp.4 SQL
Chp.6 关系数据库模式设计
Chp.11 并发控制
Chp.5 过程化SQL
Chp.7 数据库设计
Chp.12 数据库安全性
Chp.8 数据库索引
Chp.13 数据库完整性
* Chp.9 数据库应用开发
Chp.14 高级主题

本章主要内容

- 数据库语言
- SQL概述
- SQL DDL
- SQL DML
- 视图

SQL的组成


SQL
DDL
（模式）
Create Table/Alter Table/Drop Table：定义基本表
Create View/Drop View：视图操作
Create Index/Drop Index：索引操作
DML
（记录）
Insert
Delete
Select
Update
记录操作
DCL
（权限）
Grant
Revoke
权限管理
嵌入式SQL*

四、DML——插入/修改/删除记录

- DML
  - Insert：插入记录
  - Delete：删除记录
  - Update：修改记录
  - Select：查询记录

1、插入新记录到基本表中

- Insert Into <表名> (列名1，列名2，……，列名n)
  Values（值1，值2，……，值n）

```
Create Table Student(
   Sno  Varchar(10) Constraint PK Primary Key,
   Sname Varchar(20),
  Age Int,
  Sex Char(1) DEFAULT 'F'
)
```

例1：
Insert Into Student（Sno, Sname, Age, Sex）
Values（'s001'，'John'，21，'M'）

（1）Insert其它例子

例2：
Insert Into Student

Values（'s002'，'Mike'，21，'M'）

如果插入的值与表的列名精确匹配（顺序，类型），则可以省略列名表

例3：
Insert Into Student（sno, sname）

Values（'s003'，'Mary'）

如果列名没有出现在列表中，则插入记录时该列自动以默认值填充，若没有默认值则设为空

Sno	Sname	Age	Sex
s003	Mary		F

（2）日期数据的插入

例4：

```
Alter Table Student Add birth Date;
```

直接插入字符串，按年月日格式，支持多种分隔符

```
1. Insert Into Student
   Values('s004', 'Rose', 22, 'F', '1981/11/08');

2. Insert Into Student
Values('s005', 'Jack', 22, 'M',
str_to_date('12,08,1981', '%d, %m, %Y'));
```

使用str_to_date()函数插入，可以自定义格式

2、修改表中的数据

- Update <表名>
  Set <列名1>=<值1>，<列名2>=<值2>，
  ……
  Where <条件>
- 将符合<条件>的记录的一个或多个列设置新值

（1）Update例子

- 将学生John的性别改为'F'，年龄改为23

例1：
```
Update Student
Set sex＝ ' F ' ， age＝23
Where sname＝ ' John '
```

- 将所有学生的年龄都减1岁

例2：
```
Update Student
Set age＝age－1
```

3、删除表中的记录

- Delete From <表名>
  Where <条件>
- 将符合<条件>的记录从表中删除

例1：从数据库中删除学号为s001的学生

```
Delete From Student
Where sno = 's001'
```

例2：从数据库中删除所有的学生

```
Delete From Student
```

五、DML：查询数据

- SELECT查询结构
- SELECT基本查询
- 连接查询
- 嵌套查询
- 查询结果的连接：并、交、差

1、Select查询结构

- Select <列名表> ——指定希望查看的列
  From <表名列表> ——指定要查询的表
  Where <条件> ——指定查询条件
  Group By <分组列名表> ——指定要分组的列
  Having <条件> ——指定分组的条件
  Order By <排序列名表> ——指定如何排序

2、Select基本查询

- 查询全部记录：查询全部的学生信息
  - Select * From Student
  - *表示所有列
  - 等同于
    Select sno, sname, age, sex From Student
- 查询特定的列：查询所有学生的学号和姓名
  - Select sno, sname From Student

2、Select基本查询

- 使用别名：查询所有学生的学号和姓名
  - Select sno AS 学号, sname AS 姓名 From Student
  - 如果别名包含空格，须使用双引号
  - Select sno AS “Student Number” From Student

2、Select基本查询

- 使用表达式：查询所有学生的学号、姓名和出生年份，返回两列信息，其中一列是“学号：姓名”，另一列是出生年份
  - Select concat(sno,':', sname) AS 学生，2003－age AS 出生年份 From Student
  - 字符串表达式
  - 算术表达式
  - 函数表达式
    - Select sno, format_date(birth, '%m-%d-%Y') AS birthday From Student
    - Select Count(sno) As 学生人数 From Student

Oracle: sno || ':' || sname

MS SQL Server: sno + ':' + sname

2、Select基本查询

- 检索特定的记录：查询20岁以上的学生的学号和姓名
  - Select sno AS 学号, sname AS 姓名 From Student Where age > 20
  - 无Where子句时返回全部的记录
  - WHERE子句中的关系运算符
    - 算术比较符：>, <, >=, <=, =, <>
    - IN
    - IS NULL和IS NOT NULL
    - LIKE
    - EXISTS

2、Select基本查询

- IN：查询‘s001’,’s003’,’s006’和’ s008’四学生的信息
  - Select * From Student
    Where sno IN (‘s001’,’s003’,’s006’,’s008’)
- IS [NOT] NULL：查询缺少年龄数据的学生
  - Select * From Student Where age IS NULL
- LIKE：查询姓名的第一个字母为‘R’的学生
  - Select * From Student Where sname LIKE ‘R%’
  - %：任意长度的字符串
  - _：单个字符
  - 查询姓名的第一个字母为’R’并且倒数第二个字母为’S’的学生
  - Select * From Student Where sname LIKE ’R%S_’
- 多个比较式可用NOT、AND和OR连接
  - Select * From Student
    Where age IS NULL and sname LIKE ‘R%’

2、Select基本查询

- 去除重复记录：查询学生的姓名
  - Select Distinct sname From Student
  - Distinct只对记录有效，不针对某个特定列
    - Select Distinct sname, age From Student
- 排序查询结果：
  - 查询所有学生信息并将结果按年龄升序排列
  - Select * From Student Order By age
  - 将结果按年龄升序排列,按姓名降序排列
  - Select * From Student
    Order By age ASC, sname DESC
  - ASC表示升序，DESC表示降序

2、Select基本查询

- 使用聚集函数
  - Count(列名)：对一列中的值计数
  - Count(*)：计算记录个数
  - SUM(列名)：求一列值的总和（数值）
  - AVG (列名)：求一列值的平均值
  - MIN (列名)：求一列值的最小值
  - MAX (列名)：求一列值的最大值

2、Select基本查询

- 聚集函数例子
  - 求学生的总人数
    - Select count(*) From student
  - 求选修了课程的学生人数
    - Select count(distinct sno) From SC
  - 求学生的平均年龄
    - Select avg(age) as average_age From student
- 单独使用聚集函数时（Select子句中的列名都是聚集函数形式），表示对所有记录进行聚集

2、Select基本查询

- 聚集函数和分组操作：
  - 聚集函数：MIN, MAX, SUM, AVG, COUNT
  - 聚集函数一般与分组操作一起使用$\gamma_L(R)$
  - 查询男生和女生的平均年龄
    - Select sex, AVG(age) as Average_age From Student
      Group By sex

*除聚集函数外的属性必须全部出现在Group By子句中

2、Select基本查询

- 返回满足特定条件的分组结果
  - 查询不同年龄的学生人数，并返回人数在5人以上的结果
    - Select age, COUNT(*) as students From Student
      Group By age
      Having COUNT(*) > 5
  - Having子句中必须聚集函数的比较式，而且聚集函数的比较式也只能通过Having子句给出
  - Having中的聚集函数可与Select中的不同
  - 查询人数在60以上的各个班级的学生平均年龄
    - Select class, AVG(age) From Student
      Group By class
      Having COUNT(*) > 60

3、连接查询

- 一个查询从两个表中联合数据
- 返回两个表中与联接条件相互匹配的记录，不返回不相匹配的记录

Student表

Sno	Sname	Age
01	Sa	20
02	Sb	21
03	sc	21

SC表(sno是外键, cno是外键)

Sno	Cno	Score
01	C1	80
01	C2	85
02	C1	89

Course表

cno	Cname	credit
C1	Ca	3
C2	Cb	4
C3	Cc	3.5

（1）连接查询例子

- 查询学生的学号，姓名和所选课程号
  - Select student.sno, student.sname,sc.cno
    From student,sc
    Where student.sno = sc.sno ——连接条件
- 若存在相同的列名，须用表名做前缀
- 查询学生的学号，姓名，所选课程号和课程名
  - Select student.sno,
    student.sname,sc.cno,course.cname
    From student,sc,course
    Where student.sno = sc.sno and sc.cno = course.cno ——连接条件

（2）使用表别名

- 查询姓名为'sa'的学生所选的课程号和课程名
  - Select b.cno, c.cname
    From student a, sc b, course c
    Where a.sno=b.sno and b.cno=c.cno and
    a.sname='sa'
  - 表别名可以在查询中代替原来的表名使用
- 连接查询与基本查询结合：查询男学生的学号，姓名和所选的课程数，结果按学号升序排列
  - Select a.sno, b.sname, count(b.cno) as c_count
    From student a, sc b
    Where a.sno = b.sno and a.sex='M'
    Group By a.sno, b.sname
    Order By student.sno

4、嵌套查询

- 在一个查询语句中嵌套了另一个查询语句
- 三种嵌套查询
  - 无关子查询
  - 相关子查询
  - 联机视图

（1）无关子查询

- 父查询与子查询相互独立，子查询语句不依赖父查询中返回的任何记录，可以独立执行
- 查询没有选修课程的所有学生的学号和姓名
  - Select sno,sname
    From student
    Where sno NOT IN (select distinct sno From sc)
  - 子查询返回选修了课程的学生学号集合，它与外层的查询无依赖关系，可以单独执行
  - 无关子查询一般与IN一起使用，用于返回一个值列表

（2）相关子查询

- 相关子查询的结果依赖于父查询的返回值
- 查询选修了课程的学生学号和姓名
  - Select sno, sname
    From student
    Where EXISTS (Select * From sc Where sc.sno= student.sno)
  - 相关子查询不可单独执行，依赖于外层查询
  - EXISTS（子查询）：当子查询返回结果非空时为真，否则为假
  - 执行分析：对于student的每一行，根据该行的sno去sc中查找有无匹配记录

（2）相关子查询

Student表

S#	Sname	Age
01	Sa	20
02	Sb	21
03	sc	21

SC表(s#是外键, c#是外键)

S#	C#	Score
01	C1	80
01	C2	85
02	C1	89

Course表

c#	Cname	credit
C1	Ca	3
C2	Cb	4
C3	Cc	3.5

- 查询选修了全部课程的学号和姓名
  - Select sno, sname
    From student
    Where NOT EXISTS
    (Select * From course Where NOT EXISTS
    (Select * From SC Where
    sno = student.sno and cno=course.cno))
  - Select sno, sname
    From student
    Where NOT EXISTS
    (Select * From course Where cno NOT IN (Select cno From SC
    Where sno = student.sno))

对于给定的一个学生，不存在某个课程是这个学生没选的

（3）联机视图

- 子查询出现在From子句中,可以和其它表一样使用
- 查询选修课程数多于2门的学生姓名和课程数
  - Select sname, count_c
    From (Select sno, count(cno) as count_c
          From sc
          Group by sno) sc2, student s
    Where sc2.sno = s.sno and count_c>2
  - Select sname, count(cno) as count_c
    From student, sc
    Where student.sno = sc.sno
    Group by sname
    having count(cno)>=2

5、查询结果的连接

- Union和Union All
- Minus
- Intersects

（1）Union和Union All

- 查询课程平均成绩在90分以上或者年龄小于20的学生学号

```
(Select sno From student where age<20)
UNION
(Select sno
   From (Select sno, AVG(score)
           From SC
         group by sno
         having avg(score)>90) SC2
)
```

- UNION操作自动去除重复记录 ——Set Union
- Union All操作不去除重复记录 ——Bag Union

（2）Minus操作：差

- 查询未选修课程的学生学号
  - (Select sno From Student)
    Minus
    (Select distinct sno From SC)

(3)Intersect操作

- 返回两个查询结果的交集
- 查询课程平均成绩在90分以上并且年龄小于20的学生学号
  - (Select sno From student where age<20)
    Intersect
    (Select sno
    From (Select sno, AVG(score)
    From SC
    group by sno
    having avg(score)>90) SC2
)

Where are we?

- 数据库语言
- SQL概述
- SQL DDL
- SQL DML
- 视图