SQL基本语法

讲解人：李鸿岐

数据查询SELECT

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

↓

$$\pi_A\left(\sigma_F(R)\right)$$

子查询

数据查询SELECT

□ 子查询(Subqueries)

- In SQL statement, a SELECT-FROM-WHERE statement is called a query block.
- A query block embedded within WHERE or HAVING clause of another query block is called a subquery or nested query.
- Subqueries may also appear in INSERT, UPDATE, and DELETE statements.

□ There three types of subquery

- A scalar subquery returns a single column and single row.
- A row subquery returns multiple columns, but again only a single row.
- A table subquery returns one or more columns and multiple rows.

数据查询SELECT

□ 案例

查询与‘赵敏’在同一个学院的所有学生姓名

①
```
SELECT dNo
FROM Student
WHERE sName=‘赵敏’;
```

假设返回结果为’03’

②
```
SELECT sName
FROM Student
WHERE dNo=‘03’;
```

```
SELECT sName
FROM Student
WHERE dNo = (SELECT dNo
             FROM Student
             WHERE sName=‘赵敏’);
```

数据查询SELECT

□ 案例

查询选修了课程名为‘矩阵论’的所有学生姓名

```
SELECT sName
FROM Student      ③
WHERE sNo IN (SELECT sNo
           ?  FROM SC         ②
              WHERE cNo = (SELECT cNo
                        ?  FROM Course   ①
                           WHERE cName=‘矩阵论’));
```

数据查询SELECT

□ 案例

查询'王兵'同学选修的所有课程名称

```
SELECT cName
FROM Course
WHERE cNo IN (SELECT cNo
              FROM SC
              WHERE sNo = (SELECT sNo
                       ?  FROM Student
                          WHERE sName=‘王兵’));
```

数据查询SELECT

查询的属性列不是所在关系的主码

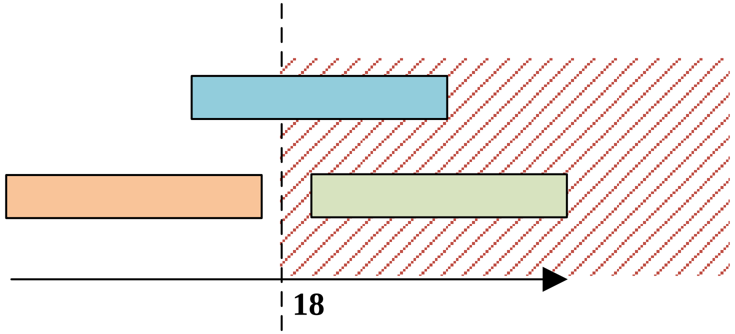
□ 案例 – 子查询包含null问题

查询不包含年龄小于18岁学生的所有学院名称

```
SELECT dName
FROM Department
WHERE dNo IN (SELECT dNo
             ? FROM Student
               WHERE age>=18);
```

逻辑替代有误

✖



sNo	sName	age	dNo
s01	赵一	19	01
s02	钱丰	16	01
s03	孙丽	16	02
s04	李响	20	03
s05	周冰	16	

□ 查询所有没选‘001’号课程的学生信息

$\Pi_{sNo}(\sigma_{cNo\neq'001'}(SC)) \bowtie Student$ ×

$(\Pi_{sNo}(Student) - \Pi_{sNo}(\sigma_{cNo='001'}(SC))) \bowtie Student$ √

数据查询SELECT

- 案例－子查询包含null问题

查询不包含年龄小于18岁学生的所有学院名称

```
SELECT dName
FROM Department
WHERE dNo NOT IN (SELECT dNo
              ? FROM Student
                WHERE age<18);
```

```
dNo NOT IN (‘01’,’02’,null);
```

null

sNo	sName	age	dNo
s01	赵一	19	01
s02	钱丰	16	01
s03	孙丽	16	02
s04	李响	20	03
s05	周冰	16	

数据查询SELECT

- □ 案例 – 子查询包含null问题

查询不包含年龄小于18岁学生的所有学院名称

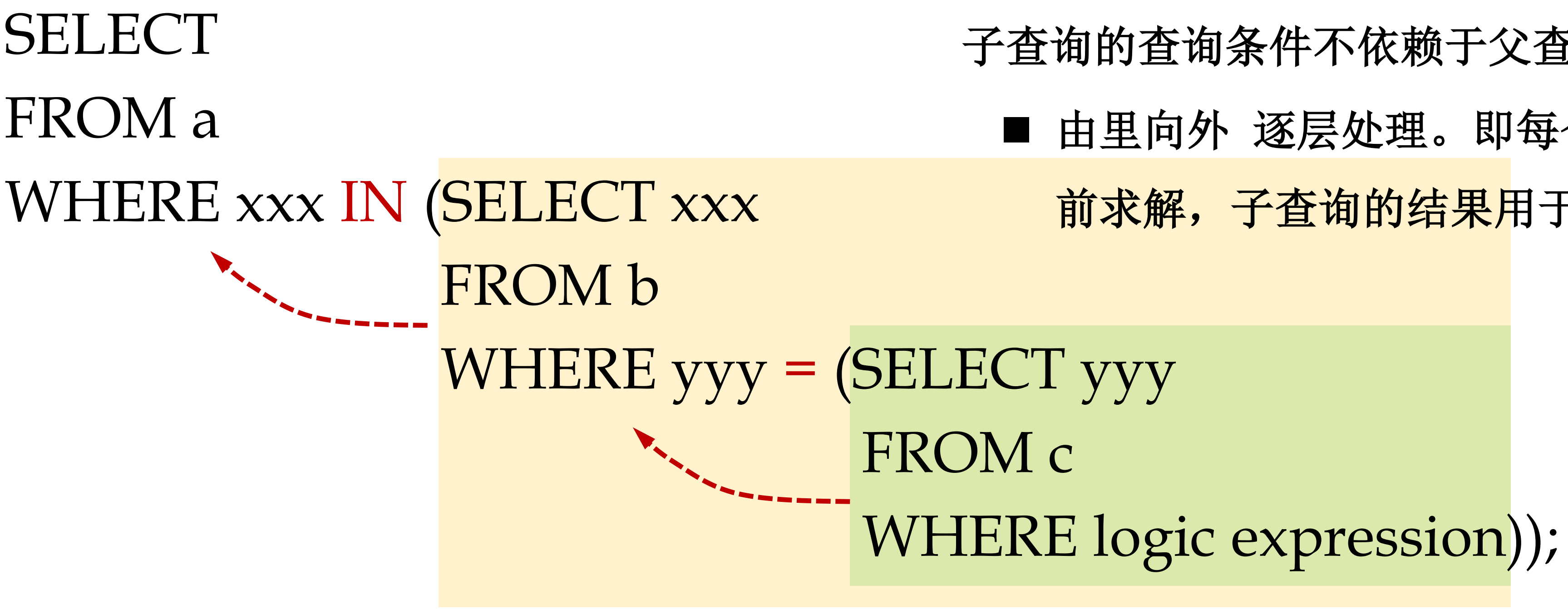
```
SELECT dName
FROM Department
WHERE dNo NOT IN (SELECT dNo
                  FROM Student
                  WHERE dNo IS NOT NULL
                    and age<18);
```

```
SELECT dName
FROM Department
WHERE (dNo NOT IN (SELECT dNo
                  FROM Student
                  WHERE age<18)) IS NOT FALSE;
```

sNo	sName	age	dNo
s01	赵一	19	01
s02	钱丰	16	01
s03	孙丽	16	02
s04	李响	20	03
s05	周冰	16	

数据查询SELECT

- □ For all the subqueries we discuss above, each subquery executes only once and the result of the subquery is used by subquery. The condition of subquery is not dependent on the condition of subquery. We call this type of query independent query.



```
SELECT
FROM a
WHERE xxx IN (SELECT xxx
              FROM b
              WHERE yyy = (SELECT yyy
                           FROM c
                           WHERE logic expression));
```

❖ 不相关子查询：

子查询的查询条件不依赖于父查询

- ■ 由里向外 逐层处理。即每个子查询在上一级查询处理之前求解，子查询的结果用于建立其父查询的查找条件。

数据查询SELECT

□ Use of comparison operators in subquery

– The subquery must place after the comparison operator.

– The subquery SELECT list must consist of a single column name or expression, except for subqueries that use the keyword EXISTS.

– The ORDER BY clause may not be used in a subquery.

```
SELECT cName
FROM Course
WHERE cNo IN (SELECT cNo
           =
          ?   FROM SC
              WHERE sNo = (SELECT sNo
                           FROM Student
                 ?         WHERE sName=‘王兵’)
          order by cNo);
```

❖ 当能确切知道内层查询返回单值时，可用比较运算符（>，<，=，>=，<=，!=或< >）。

数据查询SELECT

□ Use aggregate function in subquery

[案例]找出每个学生超过他选修课程平均成绩的课程号。

```
SELECT Sno, Cno
 FROM    SC  x
 WHERE Grade >=(SELECT AVG（Grade）
                    FROM  SC y
                    WHERE y.Sno=x.Sno);
```

相关子查询

- 相关子查询：子查询的查询条件依赖于父查询
  - ■ 首先取外层查询中表的第一个元组，根据它与内层查询相关的属性值处理内层查询，若WHERE子句返回值为真，则取此元组放入结果表
  - ■ 然后再取外层表的下一个元组
  - ■ 重复这一过程，直至外层表全部检查完为止

❖ 可能的执行过程

■ 从外层查询中取出SC的一个元组x，将元组x的Sno值（201215121）传送给内层查询。

```
SELECT AVG(Grade)
FROM SC y
WHERE y.Sno='201215121';
```

数据查询SELECT

□ Use aggregate function in subquery

查询所有年龄小于平均年龄的学生姓名

```
SELECT sNo, sName
FROM Student
WHERE age < (SELECT AVG(age)
             FROM Student);
```

□ Use ANY/ SOME and ALL in subquery

	=	<>或！ =	<	<=	>	>=
ANY	IN	--	<MAX	<=MAX	>MIN	>=MIN
ALL	--	NOT IN	<MIN	<=MIN	>MAX	>=MAX

数据查询SELECT

□ 案例

```
SELECT sName, age
FROM Student
WHERE age < ANY (SELECT age
                 FROM Student
                 WHERE dNo=’01’)
  AND dNo <> ’01’
ORDER BY age DESC;
```

```
SELECT sName, age
FROM Student
WHERE age < (SELECT MAX(age)
             FROM Student
              WHERE dNo=’01’)
  AND dNo <> ’01’
ORDER BY age DESC;
```

查询非01系中比01系任意一个学生年龄小的学生姓名和年龄

数据查询SELECT

SQL允许ANY/ALL，但不建议，转化为聚集函数去处理更明确！！

□ 案例

```
SELECT sName, age
FROM Student
WHERE age < ALL (SELECT age
                FROM Student
                WHERE dNo=’01’)
  AND dNo <> ′01′
ORDER BY age DESC;
```

```
SELECT sName, age
FROM Student
WHERE age < (SELECT MIN(age)
                FROM Student
                 WHERE dNo=’01’)
  AND dNo <> ′01′
ORDER BY age DESC;
```

查询非01系中比01系所有一个学生年龄小的学生姓名和年龄

多表查询

数据查询SELECT

□ 多表查询(Muti-Table Queries)

– To combine columns from several tables into a result table we need to use a join operation.

– To obtain information from more than one table, the choice is between using a subquery and using a join.

– If the final result table is to contain columns from different tables, then we must use a join.

查询所有学生及其选课的信息

```sql
SELECT Student.*, SC.*
FROM Student, SC
WHERE Student.sNo=SC.sNo;
```

```sql
SELECT s.sNo, s.sName, sc.cNo, sc.score
FROM Student s, sc
WHERE s.sNo=sc.sNo;
```

Note: When Cartesian product appear?

N个关系连接，WHERE至少需n-1个条件才能避免Cartesian product

数据查询SELECT

□ 案例

[案例]查询选修**2**号课程且成绩在**90**分以上的所有学生的学号和姓名。

```
SELECT Student.Sno, Sname
FROM     Student, SC
WHERE  Student.Sno=SC.Sno  AND
             SC.Cno=' 2 ' AND SC.Grade>90;
```

■执行过程:

- ●先从**SC**中挑选出**Cno='2'**并且**Grade>90**的元组形成一个中间关系
- ●再和**Student**中满足连接条件的元组进行连接得到最终的结果关系

数据查询SELECT

表与其自己连接，需要给表起别名以示区别
由于所有属性名都是同名属性，因此必须使用别名前缀

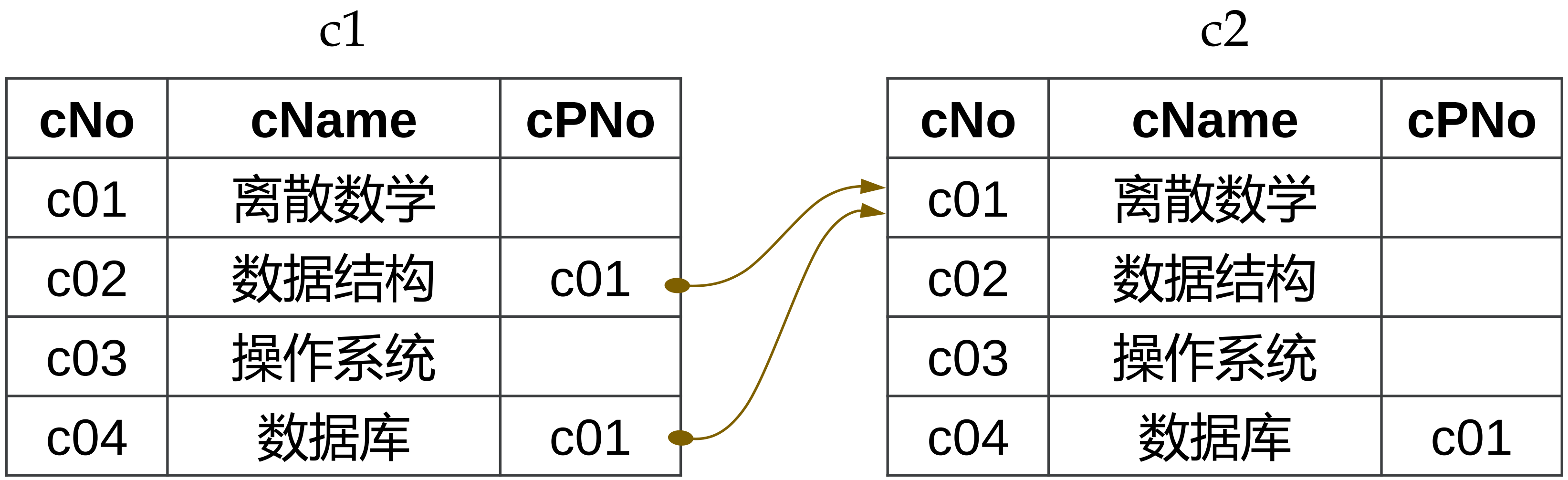
□ 多表查询(Muti-Table Queries)—自连接(self-join)

查询数据库课程先修课课程名

```
SELECT c2.cName
FROM Course c1, Course c2
WHERE c1.cPNo=c2.cNo
  and c1.cName=‘数据库’;
```

查询先修课为离散数学的课程名

```
SELECT c1.cName
FROM Course c1, Course c2
WHERE c1.cPNo=c2.cNo
  and c2.cName=‘离散数学’;
```



c1

cNo	cName	cPNo
c01	离散数学	
c02	数据结构	c01
c03	操作系统	
c04	数据库	c01

c2

cNo	cName	cPNo
c01	离散数学	
c02	数据结构	
c03	操作系统	
c04	数据库	c01

数据查询SELECT

- □ 多表查询(Muti-Table Queries)—外连接(outer join)

查询所有学生及其选课信息(包括没选课的学生)

```
SELECT s.*, sc.*
FROM Student s, SC sc
WHERE s.sNo = sc.sNo;
```

```
SELECT s.*, sc.*
FROM Student s LEFT [OUTER] JOIN SC sc  ON s.sNo = sc.sNo;
```

sNo	sName	age	dNo
s01	赵一	19	01
s02	钱丰	16	
s03	孙丽	16	02

?

sNo	cNo	score
s01	c01	
s01	c02	
s02	c01	
s02	c03	

sNo	sName	dNo	cNo	score
s01			c01	
s01			c02	
s02			c01	
s02			c03	
s03				

数据查询SELECT

□ 多表查询(Muti-Table Queries)—多表连接(multi-table join)

```
SELECT s.sName, c.cName, sc.score
FROM Student s, Course c, SC sc
WHERE s.sNo = sc.sNo   and c.cNo=sc.cNo;

SELECT s.sName, c.cName, sc.score
FROM Student s NATURAL JOIN sc JOIN Course c ON sc.cNo=c.cNo;

SELECT s.sName, c.cName, sc.score
FROM Student s, Department d, Course c, SC sc
WHERE s.dNo = d.dNo  and s.sNo=sc.sNo and c.cNo=sc.cNo
  and d.dName='软件学院';
```

NATURAL JOIN?

dno

数据查询SELECT

- 多表查询(Muti-Table Queries)—ISO SQL syntax

```
SELECT table1.column,tabel2.column
FROM  talbe1
[CROSS JOIN table2] |
[NATURAL JOIN table2] |
[JOIN table2 USING(column_name)] |
[JOIN table2
   ON(table1.column_name=table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
   ON(table1.column_name=table2.column_name)];
```

数据查询SELECT

- □ 案例

查询软件学院所有学生姓名

SELECT s.sName 子查询? √ 查询结果在同一个关系

```
SELECT s.sName
FROM Student s, Department d
WHERE s.dNo=d.dNo and d.dName=‘软件学院’;
```

查询选修了数据库系统课程的所有学生姓名及成绩

SELECT s.sName, sc.score 子查询? × 所需查询结果非同一个关系

```
SELECT s.sName, sc.score
FROM Student s, sc, Course c
WHERE s.sNo=sc.sNo and sc.cNo=c.cNo and c.cName=‘数据库系统’;
```

EXISTS/NOT EXISTS

数据查询SELECT

□ 关于EXISTS和NOT EXISTS

- EXISTS and NOT EXISTS is existential quantifiers.
- They produce a simple true/false result.
- Since EXISTS and NOT EXISTS check only for the existence or non-existence of rows in the subquery result table, the subquery can contain any number of columns.
- Usually we write the subquery as: (SELECT * FROM…)

❖ **EXISTS谓词**

- 存在量词∃
- 带有**EXISTS**谓词的子查询不返回任何数据，只产生逻辑真值“**true**”或逻辑假值“**false**”。
  - ● 若内层查询结果非空，则外层的**WHERE**子句返回真值
  - ● 若内层查询结果为空，则外层的**WHERE**子句返回假值
- 由**EXISTS**引出的子查询，其目标列表达式通常都用 *****，因为带**EXISTS**的子查询只返回真值或假值，给出列名无实际意义。

数据查询SELECT

□ 关于EXISTS和NOT EXISTS

- EXISTS and NOT EXISTS is existential quantifiers.
- They produce a simple true/false result.
- Since EXISTS and NOT EXISTS check only for the existence or non-existence of rows in the subquery result table, the subquery can contain any number of columns.
- Usually we write the subquery as: (SELECT * FROM…)

❖ **EXISTS**谓词
- ■ 存在量词 ∃
- ■ 带有**EXISTS**谓词的子查询不返回任何数据，只产生逻辑真值 “**true**”或逻辑假值 “**false**”。
  - 若内层查询结果非空，则外层的WHERE子句返回真值

❖ **NOT EXISTS**谓词
- ■ 若内层查询结果非空，则外层的**WHERE**子句返回假值
- ■ 若内层查询结果为空，则外层的**WHERE**子句返回真值

子查询只返回真值或假

数据查询SELECT

查询所有选修了’01’号课程的学生姓名

思路分析：

- 本查询涉及Student和SC关系
- 在Student中依次取每个元组的Sno值，用此值去检查SC表
- 若SC中存在这样的元组，其Sno值等于此Student.Sno值，并且其Cno= ‘01’，则取此Student.Sname送入结果表

```
SELECT Sname
 FROM Student
 WHERE EXISTS
            (SELECT *
             FROM SC
             WHERE Sno=Student.Sno AND Cno= ‘01 ');
```

数据查询SELECT

查询没有选修’01’号课程的学生姓名

```
SELECT sName
FROM Student
WHERE NOT EXISTS (SELECT *
                  FROM SC
                  WHERE sNo=Student.sNo AND cNo=‘01');
```

相关子查询 ⇨ 是否可以采用不相关子查询?

```
SELECT sName
FROM Student
WHERE sNo NOT IN (SELECT sNo
                  FROM SC
                  WHERE cNo=‘01');
```

数据查询SELECT

□ 关于EXISTS和NOT EXISTS

– Some subqueries using EXISTS or NOT EXISTS can be replace by subqueries with other form (IN or NOT IN etc.), but some can’t.

❖ 不同形式的查询间的替换

■ 一些带**EXISTS**或**NOT EXISTS**谓词的子查询不能被其他形式的子查询等价替换

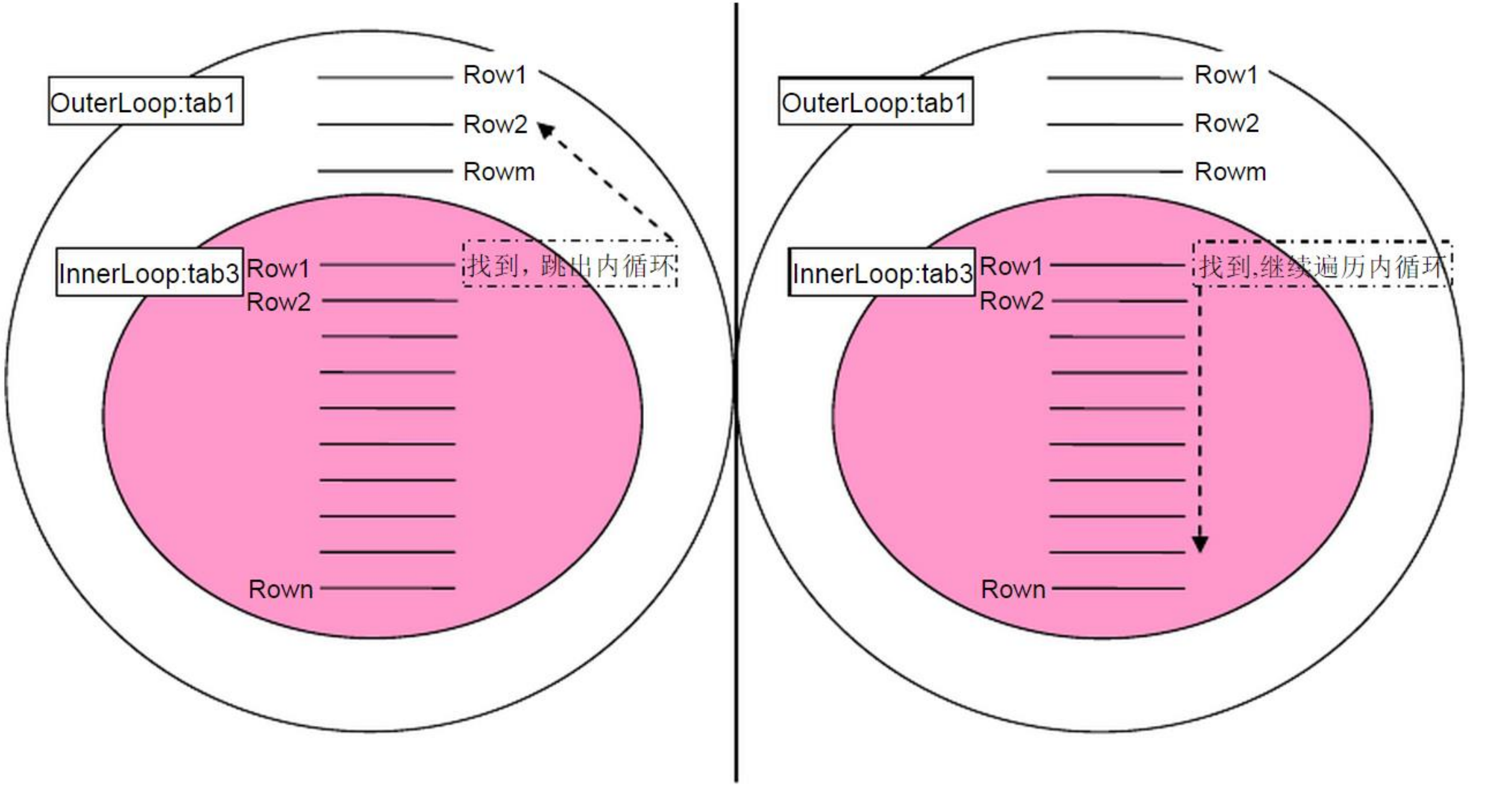
■ 所有带**IN**谓词、比较运算符、**ANY**和**ALL**谓词的子查询都能用带**EXISTS**谓词的子查询等价替换

❖ 带**EXISTS**量词的相关子查询只关心内层查询是否有返回值，并不需要查具体值

■ 效率不一定低于不相关子查询，有时是高效的方法

数据查询SELECT

- ❖ 带EXISTS量词的相关子查询只关心内层查询是否有返回值，并不需要查具体值
    - ■ 效率不一定低于不相关子查询，有时是高效的方法



(EXISTS)　　　　　　　　(IN)

数据查询SELECT

查询与赵一同学在同一个学院的学生姓名

```
SELECT sNo, sName
FROM Student S1
WHERE EXISTS (SELECT *
                        FROM Student S2
                        WHERE S2.sDept=S1.sDept
                          AND S2.sName='赵一‘);
```

相关子查询 ⇨ 是否可以采用不相关子查询?

```
SELECT sName
FROM Student
WHERE dNo = (SELECT dNo
                      FROM Student
                      WHERE sName=‘赵一’);
```

数据查询SELECT

□ 关于EXISTS和NOT EXISTS

- There is no direct expression for universal quantifier(全称量词) in SQL. We can use predication calculus to transform a predication using universal quantifiers to a predication using existential quantifiers.

$$(\forall x)P \equiv \neg(\exists x(\neg P))$$

- There is also no direct expression for implication(蕴含) in SQL. We can also tranform it to expression using existential quantifiers.

$$p \rightarrow q \equiv \neg p \vee q$$

$$(\forall y)p \rightarrow q \equiv \neg(\exists y(\neg(P \rightarrow q)))$$

$$\equiv \neg(\exists y(\neg(\neg P \vee q)) \equiv \neg \exists y(P \wedge \neg q)$$

数据查询SELECT

□ 案例

查询选修了全部课程的学生姓名

解题思路：

- ■ C中找到所有的课程数a；
- ■ SC中课程数量 b = a, 找到sNo;
- ■ S中找到学生sName。

关系代数中，使用除运算

逻辑替代有误

✖

一门课可以选多次？

□ 关于EXISTS和NOT EXISTS

– There is no direct expression for universal quantifier(全称量词) in SQL. We can use predication calculus to transform a predication using universal quantifiers to a predication using existential quantifiers.

数据查询SELECT

□ 案例

查询选修了全部课程的学生姓名

[where]没有一门课程是他不选修的

[where 1]NOT EXISTS （不存在）他没选修的课程（C）

[where 2]NOT EXISTS （不存在）该生选了该课程的元组（SC）

```
SELECT sName
FROM Student s
WHERE NOT EXISTS (SELECT *
                  FROM Course c
                  WHERE NOT EXISTS (SELECT *
                                    FROM sc
                                    WHERE sNo=s.sNo AND cNo=c.cNo));
```

数据查询SELECT

□ 案例

查询选修了‘170101’号同学所选修的全部课程的学生姓名

解题思路：

- ■ 用逻辑蕴涵表达：查询学号为**x**的学生，对所有的课程**y**，只要**170101**学生选修了课程**y**，则**x**也选修了**y**。
- ■ 形式化表示：

用**p**表示谓词 “学生**170101**选修了课程**y**”

用**q**表示谓词 “学生**x**选修了课程**y**”

则上述查询为**:** （$\forall$**y**） **p** $\rightarrow$ **q**

数据查询SELECT

□ 案例

查询选修了‘170101’号同学所选修的全部课程的学生姓名

- 等价变换：

  $(\forall y)\, p \rightarrow q \equiv \neg(\exists y(\neg(p \rightarrow q)))$

  $\equiv \neg(\exists y(\neg(\neg p \vee q)))$

  $\equiv \neg \exists y(p \wedge \neg q)$

- 变换后语义：不存在这样的课程y，学生170101选修了y，而学生x没有选。

数据查询SELECT

□ 案例

查询选修了‘170101’号同学所选修的全部课程的学生姓名

```
SELECT DISTINCT sNo
FROM SC SCX
WHERE NOT EXISTS(SELECT *
                 FROM SC SCY
                 WHERE SCY.sNo=‘170101'
                  AND NOT EXISTS(SELECT *
                                 FROM SC SCZ
                                 WHERE SCZ.sNo=SCX.sNo
                                  AND SCZ.cNo=SCY.cNo));
```

数据查询SELECT

□ 案例–行内子查询

```
SELECT sv1.sNo, s.sName, sv1.avg_score
From student s, (select sNo,avg(score) as avg_score
            from sc group by sNo)as sv1(sNo,avg_score)
Where s.sNo=sv1.sNo
```

数据查询SELECT

□ 案例–行内子查询

[案例]找出每个学生超过他自己选修课程平均成绩的课程号

```
SELECT Sno, Cno
FROM SC, (SELECT Sno, Avg(Grade)
          FROM SC
          GROUP BY Sno)
          AS   Avg_sc(avg_sno,avg_grade)
WHERE SC.Sno = Avg_sc.avg_sno
 and SC.Grade >=Avg_sc.avg_grade
```

数据查询SELECT

- □ 案例…

数据查询SELECT

1. 数据查询：SELECT：

1. 嵌套查询

不相关子查询 / 相关子查询

带 IN / 比较运算符的子查询

带有ANY / ALL的子查询

带有EXISTS / NOT EXISTS的子查询

2. 连接查询

等值连接 / 自身连接 / 外连接 / 多表连接

关于本讲内容

祝各位学习愉快！

感谢观看！

讲解人：李鸿岐