基本SQL

讲解人：陆伟 教授

SQL基本语法

- An SQL statement consists of reserved words and user-defined words.
- Most components of an SQL statement are case insensitive.
- Usually a statement terminator (;) is used to end each SQL statement. (Although the standard does not require it).
- Although SQL is free-format, an SQL statement or set of statements is more readable if indentation and lineation are used.
  - Each clause in a statement should begain on a new line.
  - The beginning of each clause should line up with the beginning of other clause.
  - If a clause has several parts, they should each appear on a separate line and be indented.
  - …

SQL基本语法

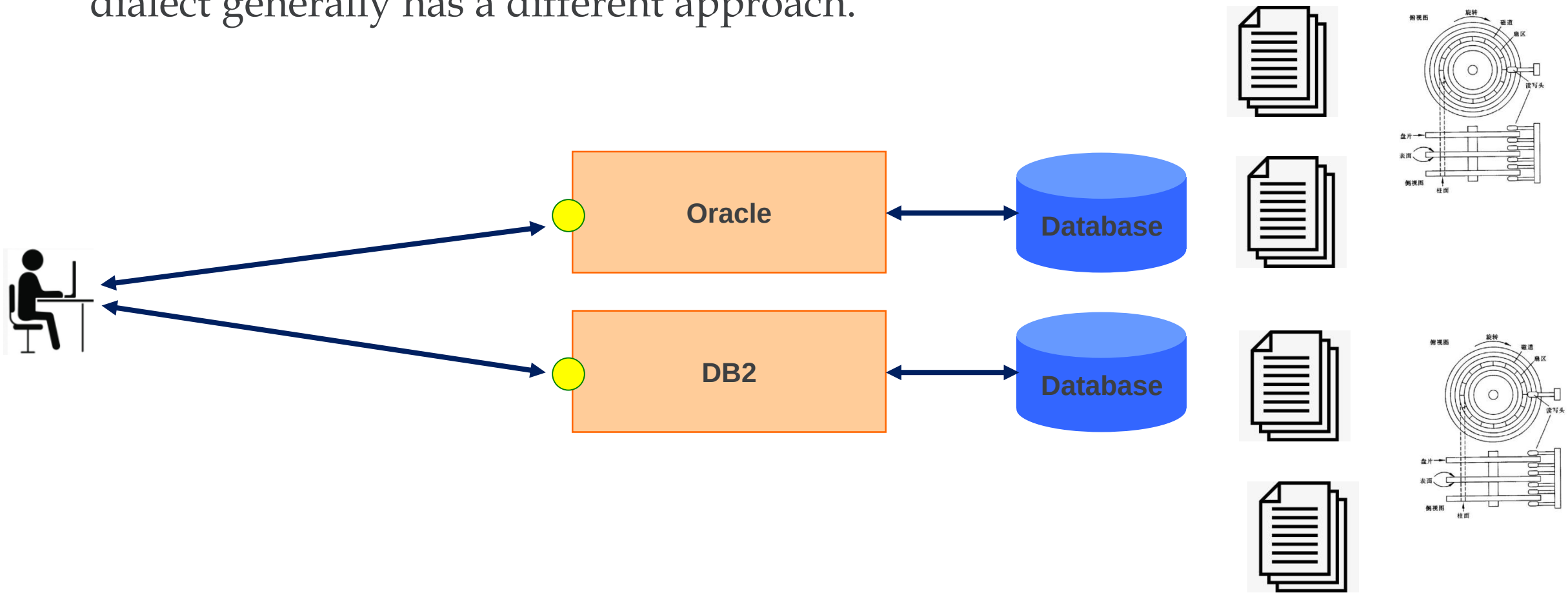
- SQL Identifiers.
  - Used to identify objects in the database, such as table names, view names, and columns
- SQL Scalar Data Types

Data type	**Declarations**
Boolean	BOOLEAN
character	CHAR VARCHAR
bit	BIT BIT VARYING
exact numeric	NUMERIC DECIMAL INTEGER SMALLINT
approximate numeric	FLOAT REAL DOUBLE PRECISION
datetime	DATE TIME TIMESTAMP
interval	INTERVAL
large objects	CHARACTER LARGE OBJECT BINARY LARGE OBJECT

- Scalar operators

数据定义语言DDL

- Creating a Database.
  - The ISO standard does not specify how database are created, and each dialect generally has a different approach.



数据定义语言DDL

- Creating a table (CREATE TABLE)

CREATE TABLE TableName
{(columnName dataType [NOT NULL][UNIQUE]
[DEFAULT defaultOption][CHECK(searchCondition)][,…]}
[PRIMARY KEY (listOfColumns),] → 实体完整性约束
{[UNIQUE (listOfColumns),][,…]}
{[FOREIGN KEY (listOfForeignKeyColumns) → 参照完整性约束
REFERENCES ParentTableName[(listOfCandidateKeyColumns)],
[MATCH {PARTIAL | FULL}
[ON UPDATE referentialAction]
[ON DELETE referentialAction]][,…]}
{[CHECK(searchCondition)][,…]}) → 用户定义完整性约束

数据定义语言DDL

Changing a Table Definition (ALTER TABLE)

ALTER TABLE TableName
[ADD [COLUMN] columnName dataType [NOT NULL][UNIQUE]
[DEFAULT defaultOption][CHECK(searchCondition)]]
[DROP [COLUMN] columnName [RESTRICT | CASCADE]]
[ADD [CONSTRAINT [constraintName]] tableConstraintDefinition]
[DROP CONSTRAINT constraintName [RESTRICT | CASCADE]]
[ALTER [COLUMN] SET DEFAULT defaultOption]
[ALTER [COLUMN] DROP DEFAULT]

Removing a Table (DROP TABLE)

DROP TABLE TableName [RESTRICT | CASCADE]

案例

关系模式

department(dNo,dName, officeRoom, homepage)
student(sNo, sName, sex, age, email, dNo)
course(cNo, cName, cPNo, credit, dNo)
sc(sNo, cNo, score, recordDate)

参考脚本：

Create Tables.sql
Insert Values.sql

数据查询SELECT

- General form

SELECT [DISTINCT | ALL]{* | [columnExpression[AS newName]][,…]}
FROM TableName[alias][,…]
[**WHERE** condition]
[GROUP BY columnList][HAVING condition]
[ORDER BY columnList]

数据查询SELECT

- 案例

```
SELECT *
FROM Student;
```

```
SELECT sNo,sName
FROM Student;
```

```
SELECT DISTINCT sNo
FROM SC;
```

```
SELECT sName, 2023-sAge
FROM Student;
```

数据查询SELECT

- Row selection (WHERE clause)

Condition	**Predicates(谓词)**
comparison	Comparison operators(=,>,<,<>,<=,>=,!=)
range	BETWEEN AND, NOT BETWEEN AND
set membership	IN, NOT IN
pattern match	LIKE, NOT LIKE
null	IS NULL, IS NOT NULL
logical operators	AND, OR,NOT

数据查询SELECT

- 案例

```
SELECT sName
FROM Student
WHERE sex = '男';
```

```
SELECT sNo,sName,dNo,age
FROM Student
WHERE age NOT BETWEEN 20 AND 23;
```

```
SELECT sName
FROM Student
WHERE age <= 17;
```

```
SELECT sName, sex
FROM Student
WHERE dNo NOT IN ('01', '02', '03');
```

数据查询SELECT

- 案例

pattern match-Regular expression：

%represents any sequence of zero or more characters (wildcard)

_represents any single character

Escape character\

```
SELECT sNo,sName,sex
FROM Student
WHERE sName LIKE ‘张%’;
```

```
SELECT *
FROM Course
WHERE cName LIKE ’DB\_%i__’ ESCAPE ’\’;
```

```
SELECT *
FROM Student
WHERE sName ~ '^[\u4E00-\u9FA5]{3,4}$';
```

数据查询SELECT

关于空值null的查询

```
SELECT sNo
FROM SC
WHERE cNo='030101' and score is null;
```

```
SELECT *
FROM Student
WHERE age NOT IN (18, 19, null);
```

逻辑运算

```
SELECT *
FROM Student
WHERE dNo='01' AND (age<20 or age>23);
```

T AND U = U　　T OR U = T
F AND U = F　　F OR U = U
U AND U = U　　U OR U = U　　NOT U = U

数据查询SELECT

- Sorting Results (ORDER BY clause)
  - In general, the rows of an SQL query result table are not arranged in any particular order. We can user ORDER BY clause in the SELECT statement to sort the results.
  - The ORDER BY clause must always be the last clause of the SELECT statement.
  - ASC—ascending order, default order
  - DESC—descending order

数据查询SELECT

- 案例

```
SELECT sNo, score
FROM SC
WHERE cNo='010101' ORDER BY score DESC;

SELECT *
FROM Student
ORDER BY dNo, age DESC;
```

Which is the place for null?

数据查询SELECT

- Using the SQL Aggregate Functions

  COUNT([DISTINCT | ALL] *)

  COUNT([DISTINCT | ALL] <columnName>)

  SUM([DISTINCT | ALL] <columnName>)
  AVG([DISTINCT | ALL] < columnName >)

  MAX([DISTINCT | ALL] < columnName >)
  MIN([DISTINCT | ALL] < columnName >)

  – COUNT, MIN, and MAX apply to both numeric and non-numeric fields, but SUM and AVG may be used on numeric field only.
  – Apart from COUNT(*), each function eliminates nulls first and operates only on the remaining non-null values.

数据查询SELECT

- 案例

```
SELECT COUNT(*)
FROM Student;
```

```
SELECT COUNT(DISTINCT sNo)
FROM SC;
```

```
SELECT COUNT(*) AS countOf
FROM Course
WHERE credit >= 2;
```

```
SELECT SUM(credit)
FROM Course
WHERE dNo=‘001’;
```

```
SELECT AVG(score)
FROM SC
WHERE cNo='010101';
```

```
SELECT MIN(age)
FROM Student
WHERE dNo='001';
```

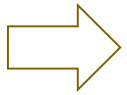
数据查询SELECT

Grouping Results (GROUP BY clause)

- All column names in the SELECT list must appear in the GROUP BY clause unless the name is used only in an aggregate function.
- When the WHERE clause is used with GROUP BY, the WHERE clause is applied first.
- The ISO standard considers two nulls to be equal for purposes of the GROUP BY clause.

```
SELECT cNo, COUNT(sNo)
FROM SC
GROUP BY cNo;
```

sNo	cNo	score
s01	001	90
s02	002	95
s01	002	80
s03	001	70
s02	003	



sNo	cNo	score	Count(sNo)
s01 s03	001	90 70	2
s01 s02	002	80 95	2
s02	003		1

数据查询SELECT

- Restricting groupings (HAVING clause)
  - HAVING clause is designed for use with the GROUP BY clause to restrict the groups that appear in the final result table.
  - The HAVING clause is not a necessary part of SQL.
  - Similar in syntax, HAVING and WHERE serve different purposes.
  - The ISO standard requires that column names used in the HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function.

```
SELECT sNo
FROM SC
WHERE score>60
GROUP BY sNo
HAVING COUNT(*)>3;
```

数据查询SELECT

- 案例

```
SELECT dNo, COUNT(sNo)
FROM Student
GROUP BY dNo
HAVING COUNT(sNo)>100;
```

```
SELECT  MAX(score), MIN(score)
FROM SC
GROUP BY cNo;
```

```
SELECT  AVG(age)
FROM Student
GROUP BY dNo;
```

```
SELECT  SUM(credit)
FROM Course
GROUP BY dNo;
```

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

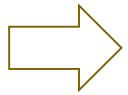
- 案例

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                                          ③
FROM Student
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

- 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);
```

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

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 supquery. The condition of subquery is not dependent on the condition of supquery. 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));
```

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 BYclause 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);
```

IN = ?

= ?

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;
```

数据查询SELECT

- 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;
```

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.

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

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

Note: When Cartesian product appear?

数据查询SELECT

- 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?

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?
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姓名及成绩

```
SELECT s.sName, sc.score      子查询?
FROM Student s, sc, Course c
WHERE s.sNo=sc.sNo and sc.cNo=c.cNo and c.cName='数据库系统';
```

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…)

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?

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.

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?

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

```
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

```
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

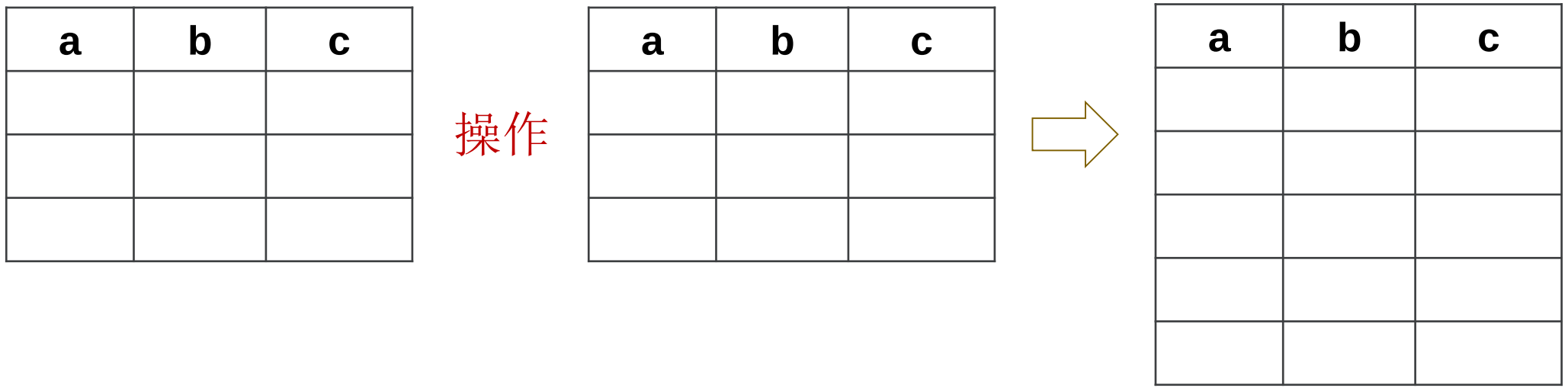
```
SELECT sv1.sNo,s.sName,sv1.avg_score
From student s, (select sNo,avg(score) as avg_score
        from sc group by sNo)as sv1(sNo,avg_score)
Where s.sNo=sv1.sNo
```

数据查询SELECT

- 案例…

数据查询SELECT

- Combining Result Tables (UNION, INTERSECT, EXCEPT)
  - In SQL, we can use the normal set operations of Union, Intersection, and Difference to combine the results of two or more queries into a single result table.
  - The three set operators in the ISO standard are called UNION, INTERSECT, and EXCEPT



数据查询SELECT

- Combining Result Tables (UNION, INTERSECT, EXCEPT)

```
(SELECT sNo
FROM SC
WHERE cNo='030101')
UNION
(SELECT sNo
FROM SC
WHERE cNo='030102');
```

```
(SELECT sNo
FROM SC
WHERE cNo='030101')
INTERSECT      EXCEPT
(SELECT sNo
FROM SC
WHERE cNo='030102');
```

数据操作语言DML

- Adding data to the database (INSERT)
  - Adds new rows of data to a table. There are two forms of INSERT statement.
  - The first allows a single row to be inserted into a named talbe.

    INSERT INTO TableName [(columnList)]
    VALUES (data ValueList)

  - The second form of the INSERT statement allows multiple rows to be copied from one or more tables to another.

    INSERT INTO TableName [(columnList)]
     SELECT …

数据操作语言DML

- 案例

```
INSERT INTO Student
VALUES ('170120', '陈冬', '男', 18, null, '01' );
```

```
INSERT INTO SC(sNo, cNo)
VALUES ('170120', '010101');
```

```
CREATE TABLE s_score(
  sNo    CHAR(6),
  sName  VARCHAR(20),
  avg    decimal(5,2));
 INSERT INTO s_score(sNo,sName,avg)
   (SELECT s.sNo,s.sName,v.avgscore
    FROM student s,(SELECT sNo,avg(score) FROM sc GROUP BY sNo)
                   as v(sNo,avgscore)
    WHERE s.sNo=v.sNo);
```

数据操作语言DML

- Modifying data in the database (UPDATE)

```
UPDATE TableName
SET columnName1=dataValue1[,columnName2=datavalue2…]
[WHERE searchCondition]
```

```
UPDATE Student
SET age=22
WHERE sNo='170101';
```

```
UPDATE Student
SET age=age+1;
```

```
UPDATE SC
SET score=60
WHERE sNo=(SELECT sNo
                FROM Student
                WHERE sName='张敏')
  and cNo=(SELECT cNo
                FROM Course
                WHERE cName='数据库系统');
```

数据操作语言DML

- Deleting data from the database (DELETE)

```
DELETE FROM TableName
[WHERE searchCondition]
```

```
DELETE FROM Student
WHERE sNo='070122';
```

```
DELETE FROM Student;
```

```
DELETE FROM SC
WHERE cNo=(SELECT cNo
                  FROM Course
                  WHERE cName='数据库系统');
```

数据操作语言DML

- 案例…

关于本讲内容

祝各位学习愉快！

感谢观看！

讲解人：陆伟 教授