SELECT **查询**

1. 使用星号和列名

SELECT \* FROM 表名

SELECT 字段名1，字段名2， …. FROM 表名

前者使用通配符 \* 是返回所有列

后者则返回指定列

2. 使用DISTINCT取消重复

使查询的字段不带重复记录

SELECT DISTINCT s\_id FROM fruits

3. 使用TOP返回前n行

SELECT TOP (4) \* FROM fruits 返回前4行

SELECT TOP 40 PERCENT \* FROM fruits 返回前40%行

tips：星号是指返回所有列，也可以把星号改为指定列

SELECT TOP (6) f\_name FROM fruits

SELECT TOP 30 PERCENT f\_name FROM fruits

4. 修改列标题

给字段取一个别名，避免与其他同名字段混淆

a. 用AS

SELECT f\_name AS fruits\_name, f\_price AS fruits\_price FROM fruits

b. 用等号(=)

SELECT fruits\_name = f\_name, fruits\_price = f\_price FROM fruits

c. 直接加别名

SELECT f\_id fruit\_id FROM fruits

tips：别名可以加单引号也可以不加

5. 在查询结果中显示字符串

为使查询结果更容易理解，可为查询的列添加一些说明性文字

SELECT 'supplier\_ID', s\_id, 'fruit\_ID', f\_id FROM fruits

tips: 字符串实质上也是一列数据，也可以像第4点一样修改列标 题

6. 查询的列为表达式

可以使用算数或逻辑运算符对查询结果进行处理

SELECT f\_name, f\_price 原价, f\_price - 1 折扣价 FROM fruits

tips：红色高亮部分也可以是f\_price \* 0.8之类

**以上6种都可以互相结合起来：**

SELECT DISTINCT TOP (5) 'fruits\_name', f\_name 水果名称, 'fruits\_price', f\_price + 100 价格, s\_id 供应商编号, f\_id 水果编号 FROM fruits

DISTINCT TOP(5) 字符串 别名 表达式结合在一条语句

WHERE**条件查询**

1. 使用关系表达式查询

关系表达式：> < = >= <= <>(不相等)

SELECT 字段名1, 字段名2, …. FROM 表名

WHERE 字段名 + 表达式

SELECT 字段名1, 字段名2, …. FROM 表名

WHERE NOT (字段名 + 表达式….)

2. 使用BETWEEN AND 表示范围

BETWEEN AND NOT BETWEEN AND

SELECT s\_id, f\_name, f\_price FROM fruits

WHERE f\_price NOT BETWEEN 5 AND 8

tips: BETWEEN AND 通常是在数字和日期比较中使用

3. 使用IN 关键字

查询指定条件范围的记录

SELECT s\_id, f\_name FROM fruits

WHERE s\_id IN (101,102,105)

SELECT s\_id, f\_id FROM fruits

WHERE s\_id NOT IN (105,102)

4. 使用LIKE 关键字

四个通配符：% \_ [] [^]

%: 匹配零和以上长度的字符串

SELECT f\_name, f\_id FROM fruits

WHERE f\_name LIKE '%a%g%'

\_: 匹配单个字符

SELECT f\_name, f\_id FROM fruits

WHERE f\_name LIKE '\_\_\_r\_'

[]: 指定范围或集合内的单个字符

SELECT f\_name, f\_id FROM fruits

WHERE f\_name LIKE '%[er]'

[^]: 不属于指定范围或集合内的单个字符

5. 使用IS NULL关键字

SELECT \* FROM customers

WHERE c\_emal IS NULL

6. 使用EXISTS关键字

EXISTS后面的参数是一个任意的子查询，如果子查询至少返回一行，EXISTS的结果为TRUE,外层查询语句进行查询，反之不查询

SELECT \* FROM productdb.dbo.fruits

WHERE EXISTS

(SELECT s\_name FROM suppliers WHERE s\_id = '107')

EXISTS也可以跟条件表达式一起用

SELECT \* FROM productdb.dbo.fruits

WHERE f\_price > 10.20 AND EXISTS

(SELECT s\_name FROM suppliers WHERE s\_id = '107')

7. ORDER BY 排序

ASC 升序 DESC 降序

SELECT f\_name FROM fruits 单列排序

ORDER BY f\_name

SELECT f\_name, f\_price FROM fruits 多列排序

ORDER BY f\_name, f\_price

默认是对数据进行升序排序，可使用DESC来降序排序

SELECT f\_name FROM fruits

ORDER BY f\_name DESC

GROUP BY

对group by用法的理解：将某个字段所有行分成值不同的组，比如

123344分成1234组，再结合聚合函数对每组对应的数据进行整合

格式：

SELECT 列名, 聚合函数1, 聚合函数2, …. FROM 表名 GROUP BY 列名

也可在函数后加一个别名，因为函数本身就是输出一个列，不加别名则函数的那一列显示”无列名”

SELECT courseDesc, COUNT(\*) FROM course

GROUP BY courseDesc

tips：关于GROUP BY 与 ORDER BY 的多列使用的理解

GROUP BY多列分组是先对前一列相同的值分组，再对后一列相 同的值分组，结合年级成绩表理解：先对班级进行分组，班级是 前一个字段，每个班都有男女生，所以再对性别进行分组，性别 是后一个字段

ORDER BY 多列排序是前一列进行排序，若前一列排序后有值相 同的行，则可以指定下一个字段进行排序，也可以结合年级成绩 表理解：将班级作为前一列，对班级进行排序后，如果1班总人 数为50，则有50行值为1的班级列，但此时这50行也即50个学生 所对应的成绩可能为乱序，这时就可以把成绩作为下一个字段进 行排序了

SELECT EmployeeID, ShipVia, Count(1) num

FROM Orders

GROUP BY EmployeeID, ShipVia

ORDER BY EmployeeID, ShipVia DESC

GROUP BY 与 ORDER BY 都是对字段按顺序进行操作

HAVING

HAVING与WHERE作用差不多，HAVING是对分组进行选择，而 WHERE是在分组之前的选择

SELECT courseDesc, SUM(courseCreadit) AS a FROM course

GROUP BY courseDesc HAVING SUM(courseCreadit) > 5.0

tips：HAVING 之后的列要是上面写的聚合函数，如果给聚合 函数列起了别名，也不能用这个别名，看上面的 SUM(courseCreadit) AS a，就不能用HAVING a > 5.0

GROUP BY 跟 HAVING 的结合使用挺重要

UNION

把两条SELECT语句连接起来，结果是把两条语句各查询的内容拼接在一块显示出来，两个表对应的列数跟数据类型要相同，顺序要一致。若UNION后不加ALL则默认不显示相同的记录，加ALL反之

SELECT s\_id 供应商编号, f\_name 水果名称, f\_price 水果价格 FROM fruits

WHERE f\_price < 9

UNION ALL

SELECT s\_id 供应商编号, f\_name 水果名称, f\_price 水果价格 FROM fruits

WHERE s\_id = 101

**聚合函数**

五个：SUM, COUNT, MAX, MIN, AVG

A. COUNT(1) 效率最高

COUNT(列名) 简单， 容易理解，效率低

COUNT(\*) 简单，占用内存大，效率相对低

**嵌套查询**

A. 使用比较运算符（返回一行一列）

> < = >= <= <>(!=)

SELECT s\_id, f\_name

FROM fruits

WHERE s\_id =

(SELECT s\_id FROM suppliers WHERE s\_city = 'Tianjin')

B. 用IN / NOT IN(内层查询语句仅返回一个数据列)

SELECT s\_name

FROM suppliers

WHERE s\_id IN

(SELECT s\_id FROM fruits WHERE f\_id = 'c0')

C. 用ANY/SOME, ALL

a. ANY跟SOME用法相同，允许对子查询的返回值列表进行比较

只要满足子查询任意一个条件，就返回一个结果作为外层查

询条件

CREATE TABLE tbl1(num1 INT NOT NULL)

CREATE TABLE tbl2(num2 INT NOT NULL)

INSERT INTO tbl1 VALUES(1), (5), (13), (27)

INSERT INTO tbl2 VALUES(6), (14), (11), (20)

SELECT \* FROM tbl1

WHERE num1 < ANY/SOME(SELECT \* FROM tbl2)

只要表tbl1的数小于表tbl2中任意一个数，则返回该数

b. ALL 需要满足子查询所有条件

D. 用EXISTS / NOT EXISTS

EXISTS后接的是子查询，如果子查询至少返回一行，则EXISTS

的结果为TRUE，进行外层循环查询

SELECT \* FROM fruits

WHERE f\_price > 10.20 AND EXISTS

(SELECT \* FROM suppliers WHERE s\_id = '107')

**连接查询**

连接查询用于主键表和外键表之间的查询。

A.内连接

a.两个表要有意义相同的两个字段A，将两个字段值用运算符(><=) 进行比较，被指定的与A字段值相关联的行都会被输出两个表的 字段A相当于一个标志，对它们进行比较。字段A就是主键与外 键相关联的那个字段（难理解）

b.内连接格式

SELECT 字段列表

FROM 表1 INNER JOIN 表2

ON 表1.字段 = 表2.字段

SELECT Employees.EmployeeID, LastName, FirstName,Orde rs.\*

FROM Employees INNER JOIN Orders

ON Employees.EmployeeID = Orders.EmployeeID

B.外连接（左右全）

a.左右连接

以左或右表为基准，与另一个表相同意义的字段比较，若被作为 基准的那个表有些行没有被另一个表匹配，则输出剩余的这些行， 且另一个表在这些行对应的值为NULL

b.全连接

全连接相当于左连接与右连接的效果加起来

c.左右全外连接格式

SELECT 字段列表

FROM 表１ LEFT/RIGHT/FULL JOIN 表2

ON 表１.字段(主键) = 表2.字段(外键)

C. 笛卡尔积结果

select \* from class,student 把两个表的数据进行组合输出

以上语句返回的结果集是class中每一条记录与student中的每一 条记录进行组合匹配，并输出。所有的记录数为class表中的行数 乘student表中的行数, 这种情况是笛卡尔积.

D. 关于内连接外连接

这两种连接实质上都是从笛卡尔积的结果中筛选出符合实际情况 的组合

内连接输出的每条两个表组合的结果都有相应的匹配，而外连接 在内连接的基础上，把基准表中没有被匹配的记录也输出来。概 括地说，内连接不会输出未被组合匹配的几录，而外连接会

E.多表查询中，如果两个表中的字段相同，并且要在查询列中使用， 则要指定是哪一个表的列。

F.表的别名

表名 别名, 表名 as 别名

G. 如果要输出多表查询中的某一个表的所有的字段，可以采用"表 名.\*"或者是"表别名.\*"

EFG代码举例

SELECT e.EmployeeID, LastName, FirstName, o.\*

FROM Employees e FULL JOIN Orders o

ON e.EmployeeID = o.EmployeeID

H.三表连接

SELECT e.EmployeeID, c.City, o.\*

FROM Orders o INNER JOIN Employees e

ON o.EmployeeID = e.EmployeeID

INNER JOIN Customers c

ON o.CustomerID = c.CustomerID

ORDER BY e.EmployeeID, c.City

I.自连接

网上用户评论的模型

A ..............

B.................

C...............

D..............

C...................

E...........

F.............

K....................

广东

广州

天河区

海珠区

增城

广西

钦州

钦南区

钦北区

南宁

code name parentCode

1 广东 null

2 广州 1

3 天河 2

4 海珠 2

5 钦州 6

create table TArea

(

code varchar(20) primary key,

name varchar(50) not null,

parentCode varchar(20) foreign key references TArea(code)

)

insert into TArea

values('44','广州',null)

insert into TArea

values('45','天河区','44')

insert into TArea

values('46','天河社区','45')

select t2.name,t1.name

from TArea t1,TArea t2

where t1.code=t2.parentCode

select t2.name,t1.name

from TArea t1 left join TArea t2

on t1.code=t2.parentCode

select tf.code, tf.name,tp.name

from TArea tf left join TArea tp

on tf.parentCode=tp.code

**查询时的注意点与技巧点**

1.当表名或字段名是某个关键字时，应该用中括号括起来，否则出 现错误

2.可以把查询的结果作为一个表

例：

SELECT 字段列表 FROM

(SELECT 字段列表 FROM 表名) AS 表名1，

(SELECT 字段列表 FROM 表名) AS 表名2

--把两个查询的结果看做表名1和表名2，即两个表

这种做法的基本结构实质上是:

SELECT 字段列表 FROM 表1， 表2

只是把表1跟表2用两个查询给替换一下而已