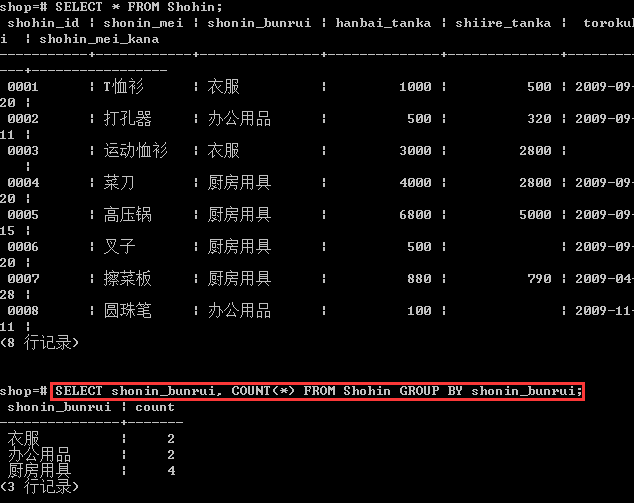
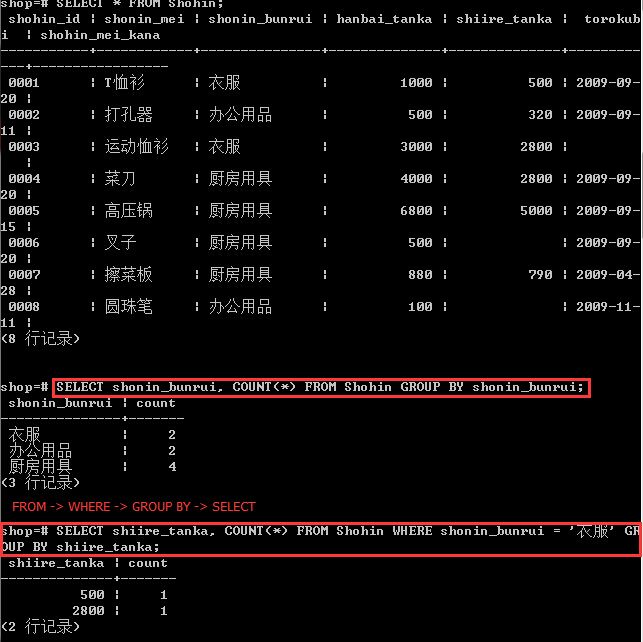
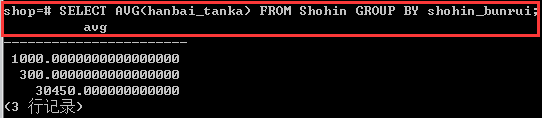
1. **GROUP BY：计算聚合结果**

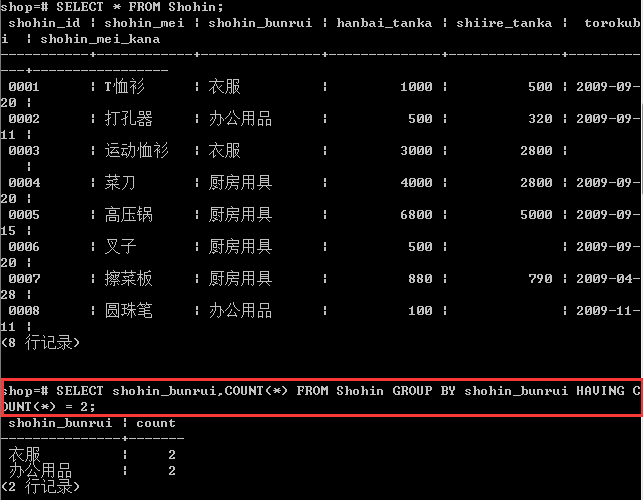




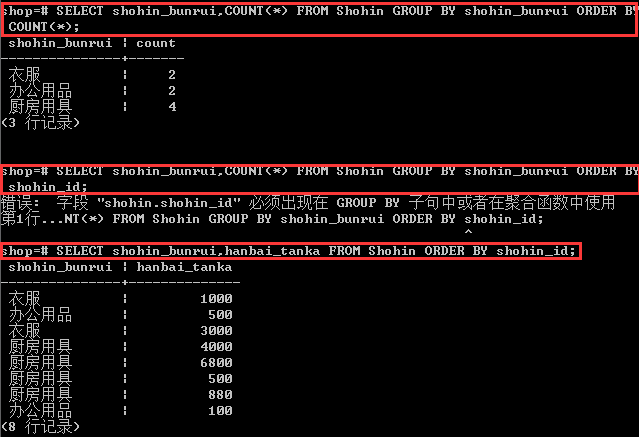
SELECT AVG(hanbai\_tanka) FROM Shohin GROUP BY shohin\_bunrui;



1. **HAVING：指定分组条件**



1. **ORDER BY：对查询结果进行排序**



1. **INSERT COPY：从一张表中选取数据，复制到另外一张表中。**

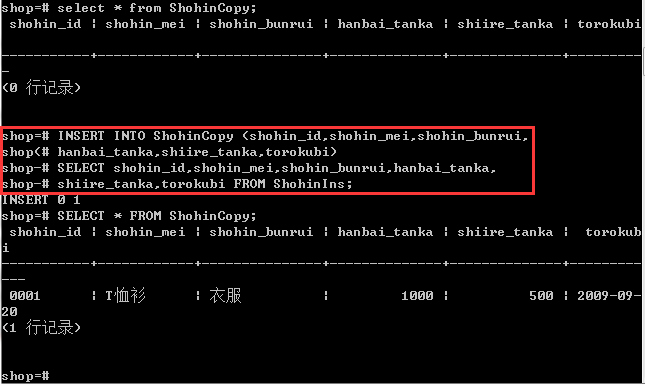
**INDERT…SELECT：**

**INSERT INTO ShohinCopy (shohin\_id,shohin\_mei,shohin\_bunrui,**

**hanbai\_tanka,shiire\_tanka,torokubi)**

**SELECT shohin\_id,shohin\_mei,shohin\_bunrui,hanbai\_tanka,**

**shiire\_tanka,torokubi FROM ShohinIns;**

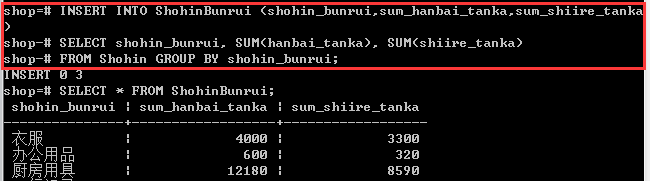


1. **INSERT…SELECT…GROUP BY**

**INSERT INTO ShohinBunrui (shohin\_bunrui,sum\_hanbai\_tanka,sum\_shiire\_tanka)**

**SELECT shohin\_bunrui, SUM(hanbai\_tanka), SUM(shiire\_tanka)**

**FROM Shohin GROUP BY shohin\_bunrui;**



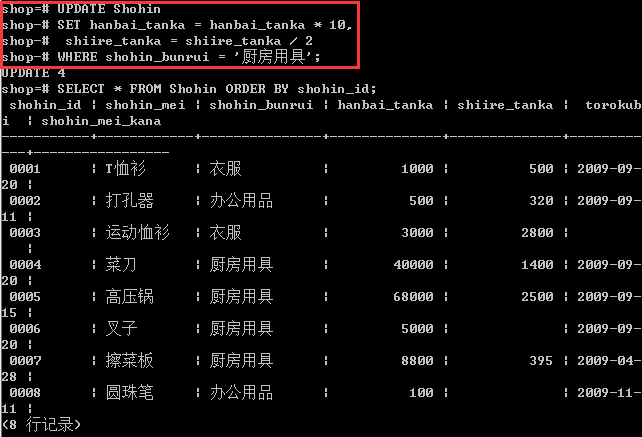
1. **UPDATE语句**

**UPDATE Shohin**

**SET hanbai\_tanka = hanbai\_tanka \* 10,**

**shiire\_tanka = shiire\_tanka / 2**

**WHERE shohin\_bunrui = '厨房用**



1. **事务：事务是需要在同一个处理单元中执行的一系列更新处理的集合。**
2. **创建事务**
3. **COMMIT提交处理**

**BEGIN TRANSACTION;**

**-- 将运动T恤的销售单价降低1000日元**

**UPDATE Shohin**

**SET hanbai\_tanka = hanbai\_tanka - 1000**

**WHERE shohin\_mei = '运动恤衫';**

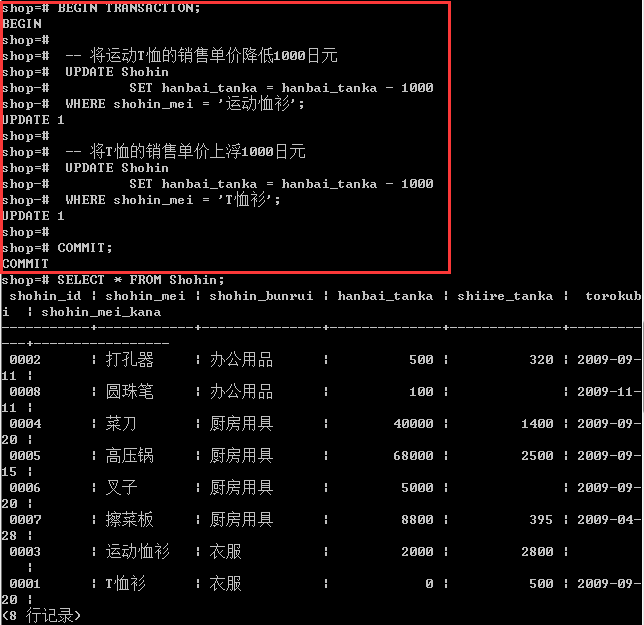
**-- 将T恤的销售单价上浮1000日元**

**UPDATE Shohin**

**SET hanbai\_tanka = hanbai\_tanka - 1000**

**WHERE shohin\_mei = 'T恤衫';**

**COMMIT;**



1. **ROLLBACK取消处理**

**BEGIN TRANSACTION;**

**-- 将运动T恤的销售单价降低1000日元**

**UPDATE Shohin**

**SET hanbai\_tanka = hanbai\_tanka - 1000**

**WHERE shohin\_mei = '运动恤衫';**

**-- 将T恤的销售单价上浮1000日元**

**UPDATE Shohin**

**SET hanbai\_tanka = hanbai\_tanka - 1000**

**WHERE shohin\_mei = 'T恤衫';**

**ROLLBACK;**



1. **ACID特性**
2. **原子性：原子性是指在事务结束时，其中所包含的更新处理要么全部执行，要么全部不执行。**
3. **一致性（完整性）：一致性指的是事务中包含的处理，要满足数据库提交前设置的约束，如主键约束或者NOT NULL约束等。**
4. **隔离性：隔离性指的是保证不同事务之间互不干扰的特性。**
5. **持久性（耐久性）：持久性指的是事务一旦结束，DBMS会保证改时点的数据状态得以保存的特性。**
6. **视图**
7. **视图的优点**
8. **表中存储的是实际数据，而视图中保存的是从表中取出数据所使用的 SELECT 语句**
9. **应该将经常使用的 SELECT 语句做成视图**
10. **创建视图**

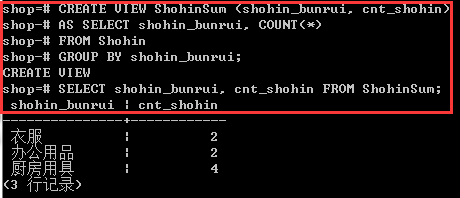
**CREATE VIEW ShohinSum (shohin\_bunrui, cnt\_shohin)**

**AS SELECT shohin\_bunrui, COUNT(\*)**

**FROM Shohin**

**GROUP BY shohin\_bunrui;**

**SELECT shohin\_bunrui, cnt\_shohin FROM ShohinSum;**



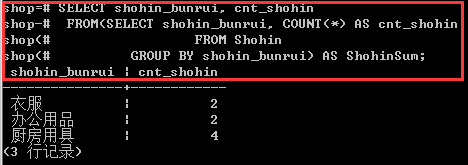
1. **视图的限制**
2. **定义视图时不能使用ORDER BY子句**
3. **视图和表需要同时进行更新，因此通过聚合得到的视图无法进行更新**
4. **子查询：子查询就是将用来定义视图的 SELECT 语句直接用于 FROM 字句当中。**

**SELECT shohin\_bunrui, cnt\_shohin**

**FROM(SELECT shohin\_bunrui, COUNT(\*) AS cnt\_shohin**

**FROM Shohin**

**GROUP BY shohin\_bunrui) AS ShohinSum;**



**多层子查询嵌套：**

**SELECT shohin\_bunrui, cnt\_shohin**

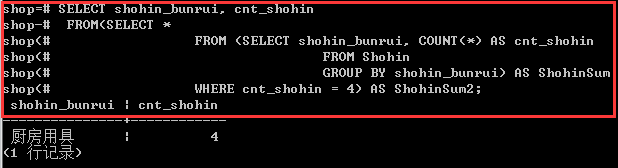
**FROM(SELECT \***

**FROM (SELECT shohin\_bunrui, COUNT(\*) AS cnt\_shohin**

**FROM Shohin**

**GROUP BY shohin\_bunrui) AS ShohinSum**

**WHERE cnt\_shohin = 4) AS ShohinSum2;**

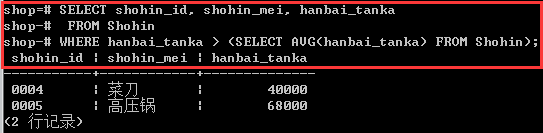


**标量子查询：必须而且只能返回一行一列的结果。**

**SELECT shohin\_id, shohin\_mei, hanbai\_tanka**

**FROM Shohin**

**WHERE hanbai\_tanka > (SELECT AVG(hanbai\_tanka) FROM Shohin);**



**关联子查询：**

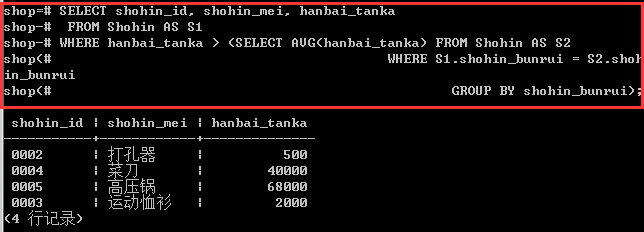
**SELECT shohin\_id, shohin\_mei, hanbai\_tanka**

**FROM Shohin AS S1**

**WHERE hanbai\_tanka > (SELECT AVG(hanbai\_tanka) FROM Shohin AS S2**

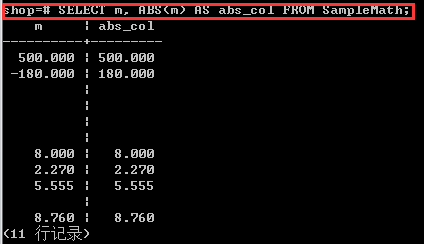
**WHERE S1.shohin\_bunrui = S2.shohin\_bunrui**

**GROUP BY shohin\_bunrui);**



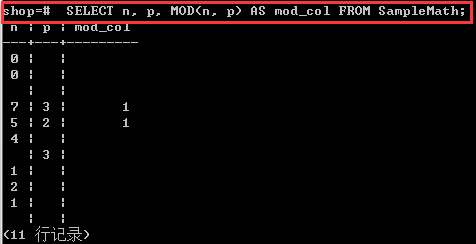
1. **算数函数**
2. **ABS()：绝对值函数**

**SELECT m, ABS(m) AS abs\_col FROM SampleMath;**



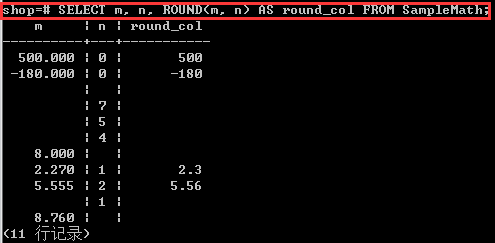
1. **MOD()：求余，SQL SERVER使用“%”运算符来计算余数**

**SELECT n, p, MOD(n, p) AS mod\_col FROM SampleMath;**



1. **ROUND()：四舍五入**

**SELECT m, n, ROUND(m, n) AS round\_col FROM SampleMath;**



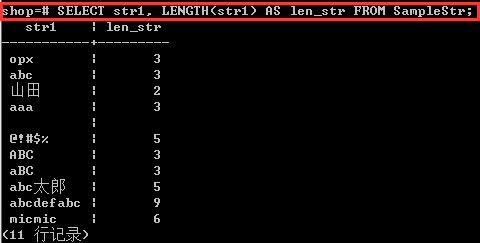
1. **字符串函数**
2. **||：拼接（SQL Server 使用“+”运算符来连接字符串； MySql使用CONCAT()函数来完成字符串的拼接）**

**SELECT str1, str2, str1 || str2 AS str\_concat FROM SampleStr;**



1. **LENGTH()：字符串长度，SQL Servrt使用LEN()函数来计算字符串的长度**

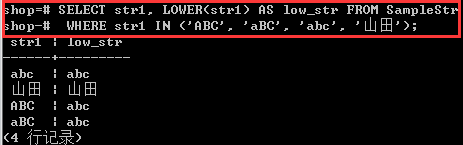
**SELECT str1, LENGTH(str1) AS len\_str FROM SampleStr;**



1. **LOWER()：小写转换**

**SELECT str1, LOWER(str1) AS low\_str FROM SampleStr**

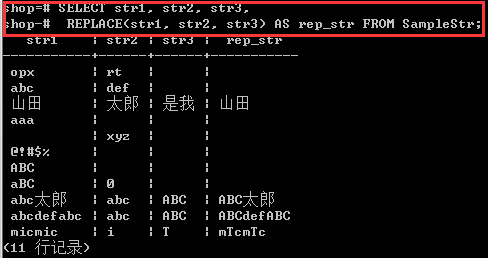
**WHERE str1 IN ('ABC', 'aBC', 'abc', '山田');**



1. **PEPLACE(对象字符串，替换前的字符串，替换后的字符串)：字符串的替换**

**SELECT str1, str2, str3,**

**REPLACE(str1, str2, str3) AS rep\_str FROM SampleStr;**

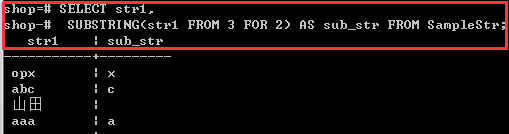


1. **SUBSTRING(对象字符串 FROM 截取的起始位置 FOR 截取的字符数)：字符串的截取。只有PostgreSQL和MySQL支持该语法。**

**SQL Server： SUBSTRING(对象字符串 截取的起始位置 截取的字符数)**

**Oracle： SUBSTR(对象字符串 截取的起始位置 截取的字符数)**

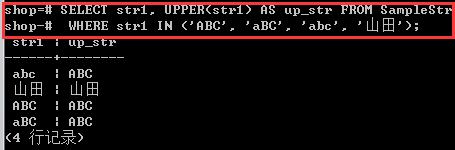
**SELECT str1, SUBSTRING(str1 FROM 3 FOR 2) AS sub\_str FROM SampleStr;**



1. **UPPER()：大写转换**

**SELECT str1, UPPER(str1) AS up\_str FROM SampleStr**

**WHERE str1 IN ('ABC', 'aBC', 'abc', '山田');**



1. **日期函数**
2. **CURRENT\_DATE：当前日期，**

**SQL Server使用CURRENT\_TIMESTAMP来获得当前日期，**

**SELECT CAST(CURRENT\_TIMESTAMP AS DATE) AS CUR\_DATE;**

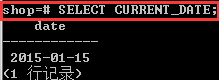
**在Oracle中需要在FROM子句中指定临时表（DUAL），**

**SELECT CURRENT\_DATE FROM dual;**

**而在DB2中需要在CRUUENT和DATE中添加空格，并需要指定临时表SYSIBM.SYSDUMMY1，**

**SELECT CURRENT DATE FROM SYSIBM.SYSDUMMY1;**

**SELECT CURRENT\_DATE;**



1. **CURRENT\_TIME：当前时间**

**SQL Server使用CURRENT\_TIMESTAMP来获得当前日期，**

**SELECT CAST(CURRENT\_TIMESTAMP AS TIME) AS CUR\_TIME;**

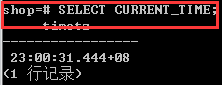
**在Oracle中需要在FROM子句中指定临时表（DUAL），**

**SELECT CURRENT\_ TIMESTAMP FROM dual;**

**而在DB2中需要在CRUUENT和TIME中添加空格，并需要指定临时表SYSIBM.SYSDUMMY1，**

**SELECT CURRENT TIME FROM SYSIBM.SYSDUMMY1;**

**SELECT CURRENT\_TIME;**



1. **CURRENT\_TIMESTAMP：当前日期和时间**

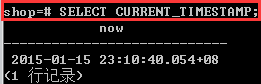
**在Oracle中需要在FROM子句中指定临时表（DUAL），**

**SELECT CURRENT\_ TIMESTAMP FROM dual;**

**而在DB2中需要在CRUUENT和TIME中添加空格，并需要指定临时表SYSIBM.SYSDUMMY1，**

**SELECT CURRENT TIME FROM SYSIBM.SYSDUMMY1;**

**SELECT CURRENT\_TIMESTAMP;**



1. **EXTRACT()：截取日期元素**

**SELECT CURRENT\_TIMESTAMP,**

**EXTRACT(YEAR FROM CURRENT\_TIMESTAMP) AS year,**

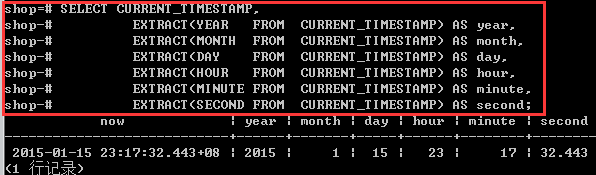
**EXTRACT(MONTH FROM CURRENT\_TIMESTAMP) AS month,**

**EXTRACT(DAY FROM CURRENT\_TIMESTAMP) AS day,**

**EXTRACT(HOUR FROM CURRENT\_TIMESTAMP) AS hour,**

**EXTRACT(MINUTE FROM CURRENT\_TIMESTAMP) AS minute,**

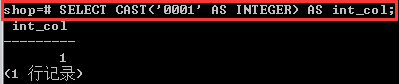
**EXTRACT(SECOND FROM CURRENT\_TIMESTAMP) AS second;**



**SQL Server、Oracle和DB2跟上面的情况一样。**

1. **转换函数**
2. **CAST(转换前的值 AS 想要转换的数据类型)：转换函数**

**SELECT CAST('0001' AS INTEGER) AS int\_col;**

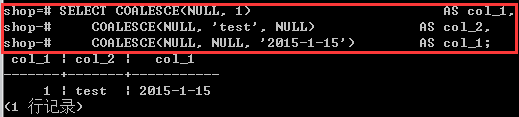


1. **COALESCE(数据1，数据2，数据3...)：将NULL转换为其他值**

**SELECT COALESCE(NULL, 1) AS col\_1,**

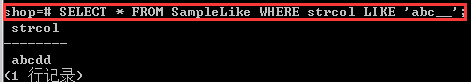
**COALESCE(NULL, 'test', NULL) AS col\_2,**

**COALESCE(NULL, NULL, '2015-1-15') AS col\_1;**



1. **谓词**
2. **LIKE谓词：字符串的部分一致查询**
3. **前方一致：“DD%”**
4. **中间一致：“%DD%”**
5. **后方一致：“%DD”**
6. **“\_”：代替“%”，代表任意一个字符**

**SELECT \* FROM SampleLike WHERE strcol LIKE 'abc\_\_';**

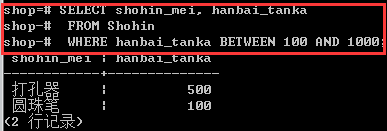


1. **BETWEENT谓词：查询范围**

**SELECT shohin\_mei, hanbai\_tanka**

**FROM Shohin**

**WHERE hanbai\_tanka BETWEEN 100 AND 1000;**

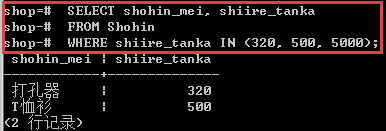


1. **IS NULL、IS NOT NULL：判断是否为空**
2. **IN谓词：OR的简便用法**

**SELECT shohin\_mei, shiire\_tanka**

**FROM Shohin**

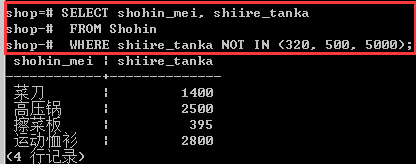
**WHERE shiire\_tanka IN (320, 500, 5000);**



**SELECT shohin\_mei, shiire\_tanka**

**FROM Shohin**

**WHERE shiire\_tanka NOT IN (320, 500, 5000);**



1. **使用子查询作为IN谓词的参数**

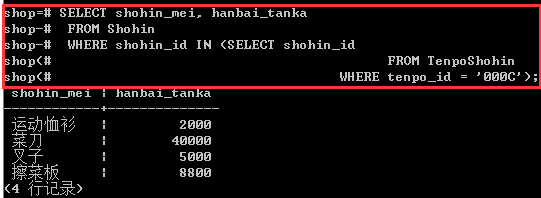
**SELECT shohin\_mei, hanbai\_tanka**

**FROM Shohin**

**WHERE shohin\_id IN (SELECT shohin\_id**

**FROM TenpoShohin**

**WHERE tenpo\_id = '000C');**



1. **EXIST谓词**

**SELECT shohin\_mei, hanbai\_tanka**

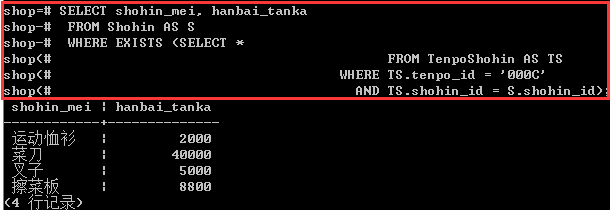
**FROM Shohin AS S**

**WHERE EXISTS (SELECT \***

**FROM TenpoShohin AS TS**

**WHERE TS.tenpo\_id = '000C'**

**AND TS.shohin\_id = S.shohin\_id);**



1. **CASE表达式**
2. **CASE表达式的语法**
3. **简单CASE表达式**

**SELECT shohin\_mei,**

**CASE shohin\_bunrui**

**WHEN '衣服' THEN 'A:' || shohin\_bunrui**

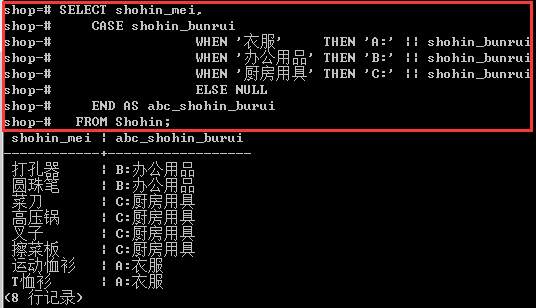
**WHEN '办公用品' THEN 'B:' || shohin\_bunrui**

**WHEN '厨房用具' THEN 'C:' || shohin\_bunrui**

**ELSE NULL**

**END AS abc\_shohin\_burui**

**FROM Shohin;**



1. **搜索CASE表达式**

**SELECT shohin\_mei,**

**CASE WHEN shohin\_bunrui = '衣服'**

**THEN 'A:' || shohin\_bunrui**

**WHEN shohin\_bunrui = '办公用品'**

**THEN 'B:' || shohin\_bunrui**

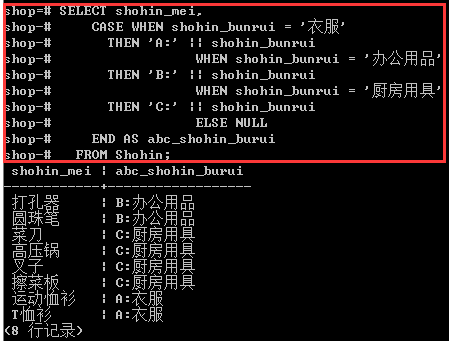
**WHEN shohin\_bunrui = '厨房用具'**

**THEN 'C:' || shohin\_bunrui**

**ELSE NULL**

**END AS abc\_shohin\_burui**

**FROM Shohin;**



1. **CASE表达式的使用方法**

**CASE表达式会从对最初WHERE子句中的<判断表达式>进行判断开始执行。如果结果我TRUE，那么就返回THEN字句中的表达式，CASE表达式的执行到此为止。如果结果不为TRUE，那么就会返回ELSE中的表达式。**

1. **表的加减法（以行为单位对表进行联结）：将满足相同规则的记录以行为单位进行连接。**
2. **表的加法：UNION，UNION运算符会去除重复的数据**

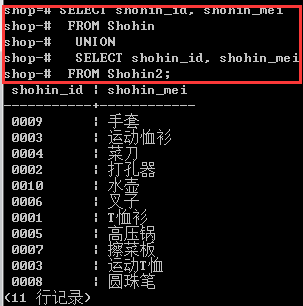
**SELECT shohin\_id, shohin\_mei**

**FROM Shohin**

**UNION**

**SELECT shohin\_id, shohin\_mei**

**FROM Shohin2;**



1. **集合运算的注意事项**
2. **作为运算对象的记录的列数必须相同**
3. **作为运算对象的记录中列的类型必须一致**
4. **可以使用任何SELECT语句，但ORDER BY字句只能在最后使用**

**SELECT shohin\_id, shohin\_mei**

**FROM Shohin**

**WHERE shohin\_bunrui = '厨房用具'**

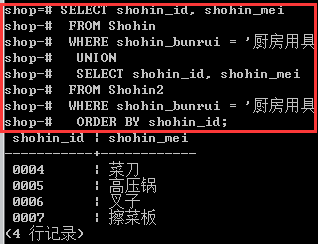
**UNION**

**SELECT shohin\_id, shohin\_mei**

**FROM Shohin2**

**WHERE shohin\_bunrui = '厨房用具'**

**ORDER BY shohin\_id;**



1. **包含重复行的集合运算：ALL选项**

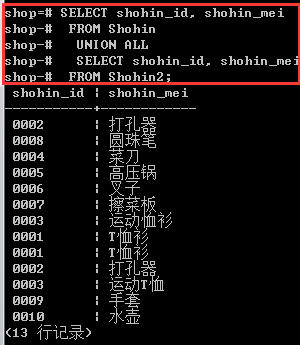
**SELECT shohin\_id, shohin\_mei**

**FROM Shohin**

**UNION ALL**

**SELECT shohin\_id, shohin\_mei**

**FROM Shohin2;**



1. **选取表中公共部分：INTERSECT，MySQL不支持**

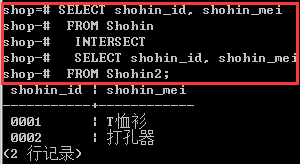
**SELECT shohin\_id, shohin\_mei**

**FROM Shohin**

**INTERSECT**

**SELECT shohin\_id, shohin\_mei**

**FROM Shohin2;**



1. **记录的减法：EXCEPT，Oracle使用MINUS进行减法运算，MySQL不支持**

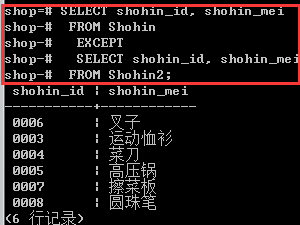
**SELECT shohin\_id, shohin\_mei**

**FROM Shohin**

**EXCEPT**

**SELECT shohin\_id, shohin\_mei**

**FROM Shohin2;**



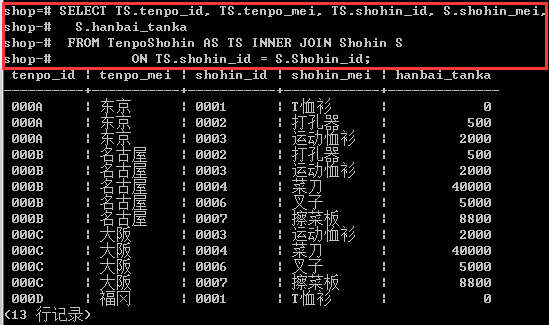
1. **联结（以列为单位对表进行联结）：就是将其他表中的列添加过来，进行“添加列”的集合运算，也可以描述为：以A表为桥梁，将B表中满足同样条件的列汇集到同一结果之中。**
2. **内联结：INNER JOIN**
3. **内联结要点a —— FROM字句：进行联结时需要在FROM字句中使用多张表**
4. **内联结要点b —— ON字句：进行内联结时必须使用ON字句，并且要书写在FROM和WHERE之间**
5. **内联结要点c —— SELECT字句：使用联结时SELECT字句中的列需要按照<表的别名>.<列名>的格式进行书写**

**SELECT TS.tenpo\_id, TS.tenpo\_mei, TS.shohin\_id, S.shohin\_mei,**

**S.hanbai\_tanka**

**FROM TenpoShohin AS TS INNER JOIN Shohin S**

**ON TS.shohin\_id = S.Shohin\_id;**



1. **外联结**
2. **外联结要点a —— 选取出单张表中全部信息**
3. **外联结要点b —— 外联结中使用LEFT、RIGHT来指定主表。使用二者所得到的结果完全相同。**

**SELECT TS.tenpo\_id, TS.tenpo\_mei, TS.shohin\_id, S.shohin\_mei,**

**S.hanbai\_tanka**

**FROM TenpoShohin AS TS RIGHT OUTER JOIN Shohin AS S**

**ON TS.shohin\_id = S.Shohin\_id;**



1. **3张表以上的联结**

**SELECT TS.tenpo\_id, TS.tenpo\_mei, TS.shohin\_id, S.shohin\_mei,**

**S.hanbai\_tanka, ZS.zaiko\_suryo**

**FROM TenpoShohin AS TS INNER JOIN Shohin AS S**

**ON TS.shohin\_id = S.shohin\_id**

**INNER JOIN ZaikoShohin AS ZS**

**ON TS.shohin\_id = ZS.shohin\_id**

**WHERE ZS.souko\_id = 'S001';**

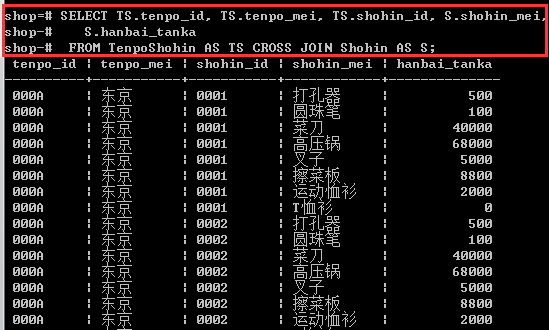


1. **交叉联结：CROSS JOIN，交叉联结的记录数是两张表中行数的乘积**

**SELECT TS.tenpo\_id, TS.tenpo\_mei, TS.shohin\_id, S.shohin\_mei,**

**S.hanbai\_tanka**

**FROM TenpoShohin AS TS CROSS JOIN Shohin AS S;**



1. **窗口函数：窗口函数也称为OLAP函数，意为对数据库数据进行实时分析处理。窗口函数就是为了实现OLAP而添加的标准SQL功能**
2. **窗口函数的语法**
3. **能够作为窗口函数的聚合函数（SUM、AVG、COUNT、MAX、MIN）**
4. **RANK、DENSE\_RANK、ROW\_NUMBER等专用窗口函数**
5. **RANK()：用来计算记录排序的函数**

**PARTITION BY能够设定排序的对象范围，即对表进行横向进行分组**

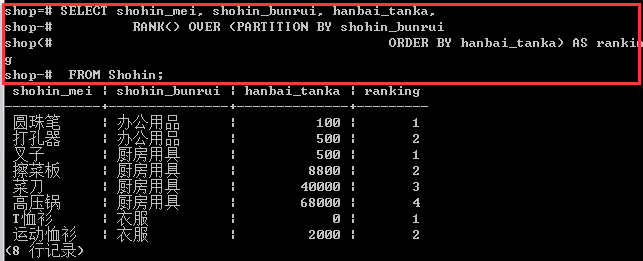
**ORDER BY能够制定按照哪一列、何种顺序进行排序，决定了纵向排序的规则**

**SELECT shohin\_mei, shohin\_bunrui, hanbai\_tanka,**

**RANK() OVER (PARTITION BY shohin\_bunrui**

**ORDER BY hanbai\_tanka) AS ranking**

**FROM Shohin;**



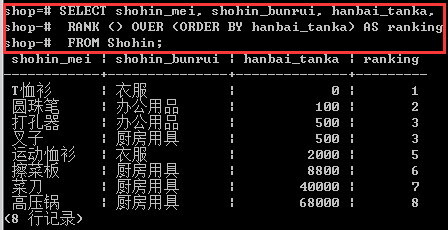
**窗口函数兼具分组和排序两种功能**

**当无指定PARTITION BY：**

**SELECT shohin\_mei, shohin\_bunrui, hanbai\_tanka,**

**RANK () OVER (ORDER BY hanbai\_tanka) AS ranking**

**FROM Shohin;**



1. **专用窗口函数的种类**
2. **RANK函数：计算排序时，如果存在同位次的记录，则会跳过之后的位次。**

**例：有3条记录排在第1位时：1位、1位、1位、4位**

1. **DENSE\_RANK函数：即使存在同样次位的记录，也不会跳过之后的位次**

**例：有3条记录排在第1位时：1位、1位、1位、2位**

1. **ROW\_NUMBER函数：赋予唯一的连续位次**

**例：有3条记录排在第1位时：1位、2位、3位、4位**

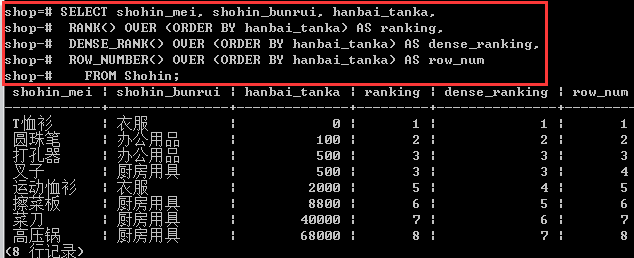
**SELECT shohin\_mei, shohin\_bunrui, hanbai\_tanka,**

**RANK() OVER (ORDER BY hanbai\_tanka) AS ranking,**

**DENSE\_RANK() OVER (ORDER BY hanbai\_tanka) AS dense\_ranking,**

**ROW\_NUMBER() OVER (ORDER BY hanbai\_tanka) AS row\_num**

**FROM Shohin;**



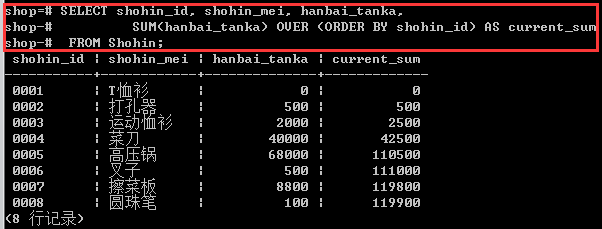
**窗口函数是对WHERE字句或者GROUP BY字句处理后的结果进行的操作，所以只能在SELECT子句中使用。**

1. **作为窗口函数使用的聚合函数**
2. **SUM**

**SELECT shohin\_id, shohin\_mei, hanbai\_tanka,**

**SUM(hanbai\_tanka) OVER (ORDER BY shohin\_id) AS current\_sum**

**FROM Shohin;**

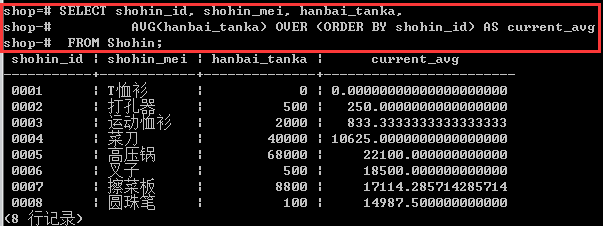


1. **AVG**

**SELECT shohin\_id, shohin\_mei, hanbai\_tanka,**

**AVG(hanbai\_tanka) OVER (ORDER BY shohin\_id) AS current\_avg**

**FROM Shohin;**



1. **计算移动平均**
2. **指定框架（统计范围）**

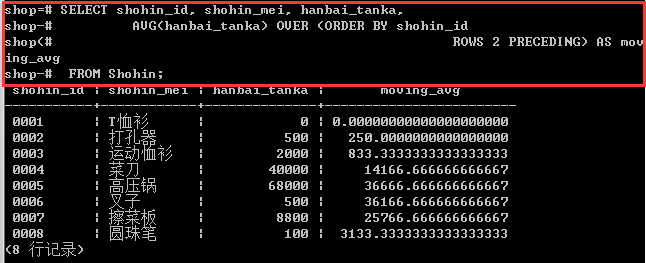
**指定最靠近 的3行作为统计对象，也就是将框架指定为截止到之前2行：**

**SELECT shohin\_id, shohin\_mei, hanbai\_tanka,**

**AVG(hanbai\_tanka) OVER (ORDER BY shohin\_id**

**ROWS 2 PRECEDING) AS moving\_avg**

**FROM Shohin;**



**将框架指定为截止到之前2行：**

1. **将当前记录的前后作为统计对象**

**将当前记录的前后行作为统计对象：**

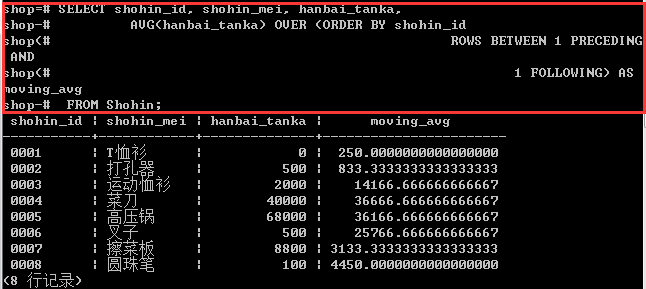
**SELECT shohin\_id, shohin\_mei, hanbai\_tanka,**

**AVG(hanbai\_tanka) OVER (ORDER BY shohin\_id**

**ROWS BETWEEN 1 PRECEDING AND**

**1 FOLLOWING) AS moving\_avg**

**FROM Shohin;**



1. **两个ORDER BY**

**OVER字句中的ORDER BY只是用来决定窗口函数按照什么样的顺序进行计算的，对结果的排列并没有影响。**

**将聚合函数作为窗口函数使用时，会以当前记录为基准来决定统计对象的记录。**

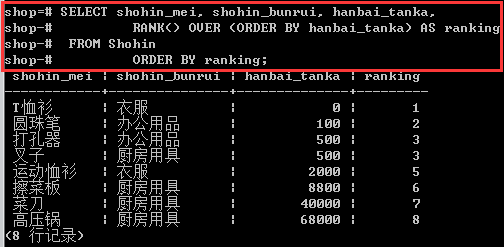
**在语句末尾使用ORDER BY字句对结果进行排序：**

**SELECT shohin\_mei, shohin\_bunrui, hanbai\_tanka,**

**RANK() OVER (ORDER BY hanbai\_tanka) AS ranking**

**FROM Shohin**

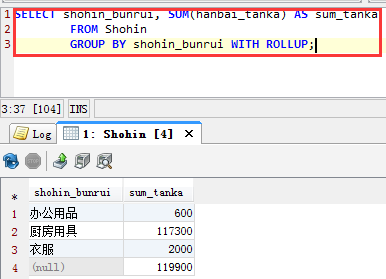
**ORDER BY ranking;**



1. **GROUPING运算符**
2. **ROLLUP：同时计算出合计值和小计值**
3. **使用ROLLUP同时计算出合计行和小计**

**MySQL：SELECT shohin\_bunrui, SUM(hanbai\_tanka) AS sum\_tanka**

**FROM Shohin GROUP BY shohin\_bunrui WITH ROLLUP;**

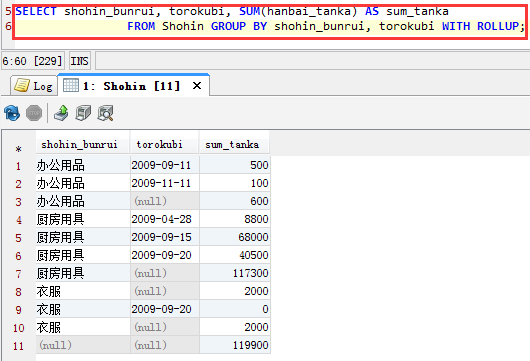


**合计行为超级分组记录，默认使用NULL作为聚合键**

1. **在GROUP BY中添加“登记日期”**

**SELECT shohin\_bunrui, torokubi, SUM(hanbai\_tanka) AS sum\_tanka**

**FROM Shohin GROUP BY shohin\_bunrui, torokubi WITH ROLLUP;**



1. **CUBE**
2. **GROUPING：该函数在其参数列的值为超级分组函数分组记录所产生的NULL时返回1，其他情况返回0**
3. **GROUPING SETS**