sql知识点总结

ch1数据库和SQL

1–1 简介

数据库（Datebase）：将大量数据保存起来，通过计算机加工而成的可以进行高效访问的数据集合。

数据库管理系统（DBMS）：database management system，用来管理数据库的计算机系统。

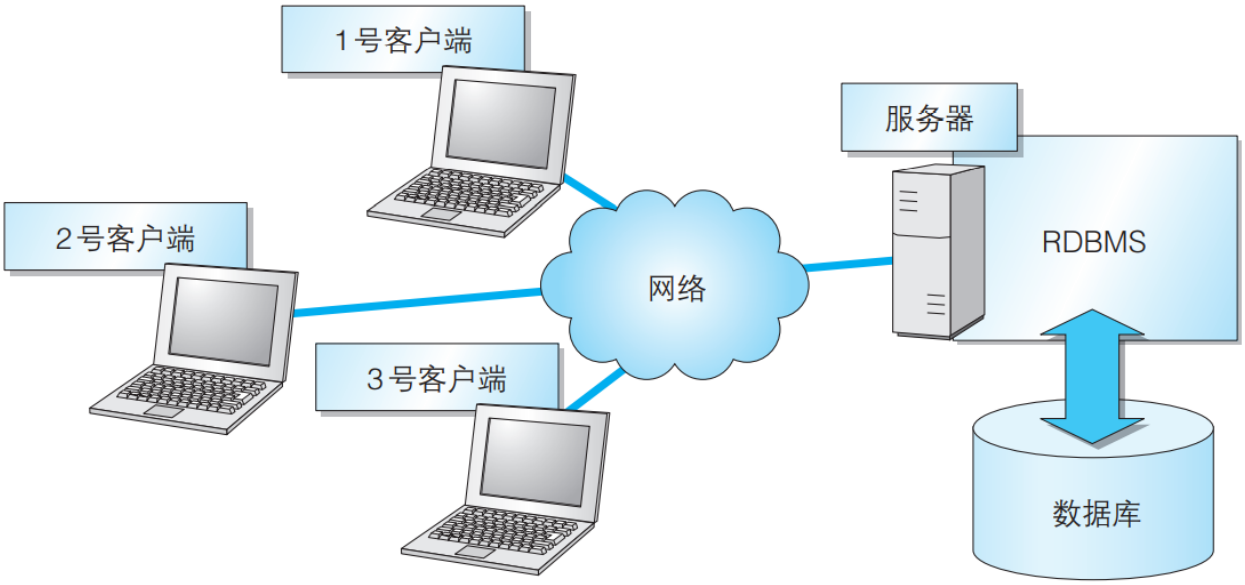
- RDBMS: 关系数据库，数据存储在利用关系互相链接的表中，每张表存储一类特定对象的数据。
  - 利用sql语言进行处理
  - 常见类型：mysql/sql server/oracle
- NoSQL：没有表或者关系，无法读取sql语言。

1–2数据库结构

RBDMS最常见结构： 客户端 / 服务器类型

- 客户端通过sql语句向服务器发送请求，服务器（RBDMS）根据该语句的内容返回所请求的数据，或者对存储在数据库中的数据进行更新。
- 多个客户端可以同时访问同一个数据库。

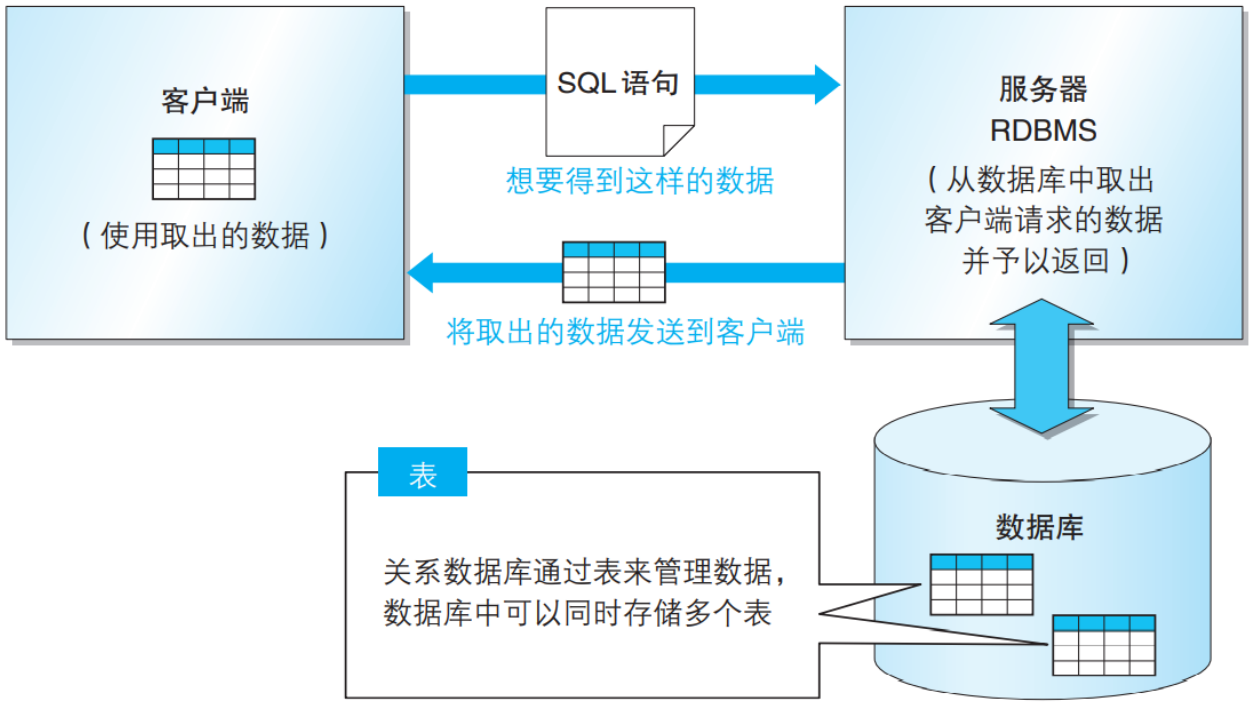
图1–4 通过网络可以实现多个客户端访问同一个数据库



- 根据 SQL 语句的内容返回的数据同样必须是二维表的形式。

1–3表的介绍

图1–5　数据库和表的关系



- 关系数据库必须以行为单位进行数据读写。

1–4语法规则

● DDL

DDL（Data Definition Language，数据定义语言）用来创建或者删除存储数据用的数据库以及数据库中的表等对象。DDL 包含以下几种指令。

CREATE：创建数据库和表等对象
DROP：　删除数据库和表等对象
ALTER：　修改数据库和表等对象的结构

● DML

DML（Data Manipulation Language，数据操纵语言）用来查询或者变更表中的记录。DML 包含以下几种指令。

SELECT：查询表中的数据
INSERT：向表中插入新数据
UPDATE：更新表中的数据
DELETE：删除表中的数据

● DCL

DCL（Data Control Language，数据控制语言）用来确认或者取消对数据库中的数据进行的变更。除此之外，还可以对 RDBMS 的用户是否有权限操作数据库中的对象（数据库表等）进行设定。DCL 包含以下几种指令。

COMMIT：　确认对数据库中的数据进行的变更
ROLLBACK：取消对数据库中的数据进行的变更
GRANT：　赋予用户操作权限
REVOKE：　取消用户的操作权限

- 关键字、表名、列名不区分大小写：本文约定，
  - 关键词大写
  - 表名开头字母大写
  - 列名等小写
- 字符串加单引号；日期固定格式，本书为 '2010-01-26' ；数字没有要求
- 单词空格需要半角空格/换行符
- 只能使用半角英文字母、数字、下划线（_）作为数据库、表和列的名称 。
- 也就是说，中文只能作为别名，select as语句里面，而且要用双引号引起来。
- 名称必须以半角英文字母开头 。
- 所有的列都必须指定数据类型。

四种基本的数据类型：

- INTEGER型 ： 存储整数

- CHAR型 ： 存储字符串（不同RDBMS使用的单位长度不一样，有的是字符个数，有的是字节）。定长字符串。
- VARCHAR型 ： 存储字符串， 以可变长字符串的形式。（ Oracle中使用VARCHAR2型， DATE型还包含时分秒 ）
- DATE型 ： 存储日期（年月日）。

约束：

- NOT NULL，非空，必须输入；只能以列为单位进行设置（不可在语句末尾全局设置）
- PRIMARY KEY (product_id) ： 用来给 product_id 列设置主键约束， 主键（primary key）就是可以唯一确定一行数据的列 。所以设定主键约束的列数据不可重复。最好设定一个主键，便于后期的增减。主键必须非null。
- 注意：主键可以设定多列，虽然单列可以重复，但是组合之后必须唯一。

1-5 表的删除和更新

- CREATE TABLE Product; 新增表
- DROP TABLE Product; 删除表
- RENAME TABLE Poduct to Product; 修改表名
- ALTER TABLE <表名> ADD COLUMN <列的定义>; 添加列
- ALTER TABLE Product ADD COLUMN product_name_pinyin VARCHAR(100);
- ALTER TABLE <表名> DROP COLUMN <列名>; 删除列
- ALTER TABLE Product DROP COLUMN product_name_pinyin;

插入数据：

- INSERT INTO Product VALUES ('0001', 'T恤衫', '衣服', 1000, 500, '2009-09-20');
- insert into 后面不用加table，因为也只能往table里面加入数据了

ch2 查询基础

2-1查询SELECT

- 使用别名：select A as C from B ;
- 别名C可以使用中文或包括空格的字符串，但需要用双引号（"）括起来 。一般建议只用单词
- 可为常数设定别名如下：

代码清单2-6　查询常数

```
SELECT '商品' AS string, 38 AS number, '2009-02-24' AS date,
       product_id, product_name
  FROM Product;
```

```
 string  |  number  |    date     | product_id | product_name
---------+----------+-------------+------------+--------------
 商品    |  38      | 2009-02-24  | 0001       | T恤衫
 商品    |  38      | 2009-02-24  | 0002       | 打孔器
 商品    |  38      | 2009-02-24  | 0003       | 运动T恤
 商品    |  38      | 2009-02-24  | 0004       | 菜刀
 商品    |  38      | 2009-02-24  | 0005       | 高压锅
 商品    |  38      | 2009-02-24  | 0006       | 叉子
 商品    |  38      | 2009-02-24  | 0007       | 擦菜板
 商品    |  38      | 2009-02-24  | 0008       | 圆珠笔
```

- 从结果中删除重复行 ： select distinct A,B from C；
- 注意：select distinct A ,B，对AB的组合进行去重而不是分别对A，B去重或只对A去重
- 在使用 DISTINCT 时，NULL 也被视为一类数据。
- 筛选行： select A from B where xxx;
- 先通过 WHERE 子句查询出符合指定条件的record，然后再选取出 SELECT 语句指定的列
- select A from B where xx limit b,a；共返回a行，从b+1行开始返回
- select A from B where xx limit a; 返回前a行
- select A from B where xx limit a offset b; 共返回a行，从b+1行开始返回

2-2 比较运算符

- 比较运算符：!=;
- 所有包含 NULL 的计算，结果都是 NULL
- NULL不能参与比较运算符，对应的真值为unknown
- 判定某列是否包含NULL，用IS NULL/IS NOT NULL;

2-3 逻辑运算符

- NOT表示否定，AND，OR
- AND 运算符优先于 OR 运算符,可以用括号加强运算顺序
- 真值就是值为真（TRUE）或假 （FALSE）其中之一的值 ，但sql里面也包括unknown
- 对于unknow的结果，返回为空（NULL）

表2-6　三值逻辑中的AND和OR真值表

AND

P	Q	P AND Q
真	真	真
真	假	假
真	不确定	不确定
假	真	假
假	假	假
假	不确定	假
不确定	真	不确定
不确定	假	假
不确定	不确定	不确定

OR

P	Q	P OR Q
真	真	真
真	假	真
真	不确定	真
假	真	真
假	假	假
假	不确定	不确定
不确定	真	真
不确定	假	不确定
不确定	不确定	不确定

- in操作符: where a in (a,b,c);功能和or一样
- in操作符一般比or快，而且括号内可以包含select语句
- 注意：IN和NOT IN都无法取出NULL，只有IS NULL才可以

2-4 where+LIKE

like是谓词，

1.谓词就是返回值为 TRUE/FALSE/UNKNOWN 的函数

2. LIKE，BETWEEN，IS NULL，IS NOT NULL，IN，EXISTS

3. BETWEEN范围查询包括临界值

百分号%：可以匹配0个至多个字符，但不可以匹配NULL

- 以a开头： where xx LIKE 'a%';
- 以a结尾： where xx LIKE '%a';
- 含有a： where xx LIKE '%a%';

下划线_: 只可匹配单个字符

- eg. where xx LIKE'_ _ inch teddy bear' 说明这个必须是2位数

方括号[ ]: mysql不支持

ch3 聚合与排序

3-1聚合函数

- 注意，下面除了count(*)以外的聚合函数，均去除了NULL数据再进行计算

SQL中 select count(1) count中的1 到底是什么意思呢?和count(*)的区别

count(1)，其实就是计算一共有多少符合条件的行。
1并不是表示第一个字段，而是表示一个固定值。
其实就可以想成表中有这么一个字段，这个字段就是固定值1，count(1)，就是计算一共有多少个1.
同理，count(2)，也可以，得到的值完全一样，count('x')，count('y')都是可以的。一样的理解方式。在你这个语句理都可以使用，返回的值完全是一样的。就是计数。
count(*)，执行时会把星号翻译成字段的具体名字，效果也是一样的，不过多了一个翻译的动作，比固定值的方式效率稍微低一些。

- count,max,min,avg,sum
- avg函数求比率！！！所需的类别为1，其余为0即可。结合case when then else end语句
- 下面二者等价：

select seller_id, sum(case when feedback_score = 1 then 1 else 0 end)/count(*) rate

from trans_fdback

group by seller_id

having rate>=0.5;

select seller_id, avg(case when feedback_score = 1 then 1 else 0 end) rate

from trans_fdback

group by seller_id

having rate>=0.5;

- 当聚合键（分组依据）中包含(多行)NULL时，将NULL作为一组特定的数据进行聚合统计
- GROUP BY子句的聚合键：检索列，或出现在select语句中的有效的表达式（有则必须用上）。可用别名。
- eg.

```sql
select substring_index(profile,',',-1) as gender,count(device_id)
from user_submit
group by gender
```

- select语句：聚合键，常数，聚合函数
- GROUP BY子句结果的显示是无序的
- 只有select、having、order by子句可以使用聚合函数
- where不可以使用聚合函数，因为它筛选的是还没有groupby的数据

3-3过滤分组：having子句

- having可以通过聚合函数对分组结果进行筛选，然后选出目标组。
- having子句的要素：聚合键（但最好在where里面限制完）、聚合函数
- `WHERE 子句 = 指定行所对应的条件`
- `HAVING 子句 = 指定组所对应的条件`

3-4查询结果排序:order子句

- 降序DESC：order by A DESC，C
- 排序键中包含NULL时，会在开头或末尾进行汇总
- （未分组）SELECT子句中未包含的列也可以在ORDER BY子句中使用
- （分组）ORDER BY子句中也可以使用聚合函数,而且可以是select里面没有出现的聚合函数
- 列编号是指 SELECT 子句中的列按照从左到右的顺序进行排列时所对应的编号（1, 2, 3, …）
- 在ORDER BY子句中可以使用列编号，但是最好不用，因为可读性不强

ch4 数据更新

4-1数据的插入INSERT

- 列清单-值清单
- 当需要对每一列进行插入时，可以省略列清单。

代码清单4-3　省略列清单

```
-- 包含列清单
INSERT INTO ProductIns (product_id, product_name, product_type,
sale_price, purchase_price, regist_date) VALUES ('0005', '高压锅',
'厨房用具', 6800, 5000, '2009-01-15');

-- 省略列清单
INSERT INTO ProductIns VALUES ('0005', '高压锅', '厨房用具',
6800, 5000, '2009-01-15');
```

- 插入NULL值，前提：对应列没有NOT NULL约束

代码清单4-4　向purchase_price列中插入NULL

```
INSERT INTO ProductIns (product_id, product_name, product_type,
sale_price, purchase_price, regist_date) VALUES ('0006', '叉子',
'厨房用具', 500, NULL, '2009-09-20');
```

- 插入默认值：前提，创建表的时候设置了default约束
- 显性插入：列清单不省略，值清单对应填写default
- 隐形插入：列清单和值清单对应位置直接不填，均省略；建议不用这种方法，不直观

- 如果列清单省略了没有设定默认值的列，该列的值就会被设定为 NULL。此时，如果省略的是设置了 NOT NULL 约束的列，INSERT 语句就会出错。
- replace into 跟 insert into功能类似，不同点在于：replace into 首先尝试插入数据到表中，如果发现表中已经有此行数据（根据主键或者唯一索引判断）则先删除此行数据，然后插入新的数据;否则，直接插入新数据。

ch5 复杂查询

5-1 子查询

1. 嵌套子查询：不关联，简单。
2. 作为计算字段使用子查询：也简单，但有更好方案

要对每个顾客执行COUNT(*)，应该将它作为一个子查询。请看下面的代码：

输入▼

```
SELECT cust_name,
       cust_state,
       (SELECT COUNT(*)
       FROM Orders
       WHERE Orders.cust_id = Customers.cust_id) AS orders
FROM Customers
ORDER BY cust_name;
```

相当于：对每一个customers里面的id，验证orders表的id等于这个id的行，全部返回，然后count(*)看一共有多少行；如果换成id=id，不指定表，就会默认是orders里面的表，同一列进行比较，返回的都是TRUE，所以返回所有列，count(*)完就是5，是一个常数。

3. 联结

5-2内联结：下面的写法没有区别，（也叫等值联结）

inner join = join

输入▼

```
SELECT vend_name, prod_name, prod_price
FROM Vendors INNER JOIN Products
  ON Vendors.vend_id = Products.vend_id;
```

输入▼

```
SELECT vend_name, prod_name, prod_price
FROM Vendors, Products
WHERE Vendors.vend_id = Products.vend_id;
```

注意：下面使用子查询和内联结作用等价

输入▼

```
SELECT cust_name, cust_contact
FROM Customers
WHERE cust_id IN (SELECT cust_id
                  FROM Orders
                  WHERE order_num IN (SELECT order_num
                                      FROM OrderItems
                                      WHERE prod_id = 'RGAN01'));
```

输入▼

```
SELECT cust_name, cust_contact
FROM Customers, Orders, OrderItems
WHERE Customers.cust_id = Orders.cust_id
  AND OrderItems.order_num = Orders.order_num
  AND prod_id = 'RGAN01';
```

5-3自联结：

5-4自然联结：

5-5外联结：

ch6 函数

6-1字符串函数:

- CONCAT(str1,str2,...),拼接
- LENGTH(x), 长度
- LEFT(x，a)取出x的左边a个字符
- LOWER(x),UPPER(x), 大小写
- REPLACE(str,fstr,tstr), 替换
- SUBSTR(str FROM pos FOR len), 截取
- TRIM(x),LTRIM(x),RTRIM(x) 去空格
- SOUNDEX(str):可用来匹配发音类似的string，eg. where SOUNDEX(name) = SOUNDEX('Michael')

6-2日期函数：

SELECT CURRENT_DATE; -- 当前日期

SELECT CURRENT_TIME; -- 当前时间

SELECT CURRENT_TIMESTAMP; -- 当前日期、时间

提取时间元素：

```
SELECT CURRENT_TIMESTAMP,
 YEAR(CURRENT_TIMESTAMP) AS year,
 MONTH(CURRENT_TIMESTAMP) AS month,
 DAY(CURRENT_TIMESTAMP) AS day,
 HOUR(CURRENT_TIMESTAMP) AS hour,
 MINUTE(CURRENT_TIMESTAMP) AS minute,
 SECOND(CURRENT_TIMESTAMP) AS second;
```

时间差：

```
timestampdiff(second, start_time, end_time)
```

datediff(out_time,in_time)=1

timestampdiff要把小的时间（更早的时间）放在前面

datediff要把大的时间（更靠后的时间）放在前面

减去某个时间：

DATE_SUB(OldDate,INTERVAL 2 DAY)

提取固定格式：

DATE_FORMAT(trans_timestamp,'%y-%m-%d')

比以字符串形式，用left什么的快

6-3转换函数（用来转换数据类型和值的函数）

- cast函数，类型转换

```
SELECT CAST('0001' AS SIGNED INTEGER) AS int_col;

SELECT CAST('2009-12-14' AS DATE) AS date_col;
```

- COALESCE，返回可变参数中左侧开始第1个不是 NULL 的值

```
SELECT COALESCE(NULL, 1) AS col_1,
 COALESCE(NULL, 'test', NULL) AS col_2,
 COALESCE(NULL, NULL, '2009-11-01') AS col_3;
```

ch7sql高级处理

7-1 窗口函数

https://zhuanlan.zhihu.com/p/120269203

1. 窗口函数只能在SELECT子句中使用。
2. partition by 指定窗口分割依据
3. order by 指定纵向计算时排序的规则，跟最终排序的order by无关
4. 窗口函数是对 WHERE子句或者GROUP BY子句处理后的“结果”进行的操作。(在得到排序结果之后，如果通过 WHERE 子句中的条件除去了某些记录，或者使用 GROUP BY 子句进行了汇总处理，那好不容易得到的排序结果也无法使用了。)
5. 不写PARTITION BY，相当于整个表作为一个大窗口
6. rank()，存在相同位次的记录，则会跳过之后的位次，很奇怪，不能使用别名为rank（奇怪报错）
7. DENSE_RANK(), 即使存在相同位次的记录，也不会跳过之后的位次
8. ROW_NUMBER(), 赋予唯一的连续位次

9. 聚合函数也可作窗口函数

移动平均汇总：

1. 本行+前两行：ROWS 2 PRECEDING

```
SELECT product_id, product_name, sale_price,
 AVG (sale_price) OVER (ORDER BY product_id ROWS 2 PRECEDING) AS moving_avg
 FROM Product;
```

2. 前一行+本行+后一行：ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING

```
SELECT product_id, product_name, sale_price,
 AVG (sale_price) OVER (ORDER BY product_id ROWS BETWEEN 1 PRECEDING AND 1
FOLLOWING)
AS moving_avg
 FROM Product;
```

7-2同时得出合计和小计： WITH ROLLUP

1. rollup单组A，group by() + group by A, 多一行汇总

```
SELECT product_type, SUM(sale_price) AS sum_price
FROM Product
GROUP BY product_type WITH ROLLUP;
```

2. rollup多组A，B, group by() + group by A + group by A,B, 多1+A行汇总

```
SELECT product_type, regist_date, SUM(sale_price) AS sum_price
 FROM Product
 GROUP BY product_type, regist_date WITH ROLLUP;
```