

课堂主题

MySQL介绍和基本的SQL编写及解析

课堂目标

- 1、理解关系型数据库的概念
- 2、了解MySQL历史
- 3、掌握MySQL在Linux下的安装及远程连接
- 4、掌握基本SQL的编写
- 5、理解SQL的解析顺序
- 6、理解多表间关联
- 7、掌握多表关联查询及子查询

MySQL介绍篇

数据库概述

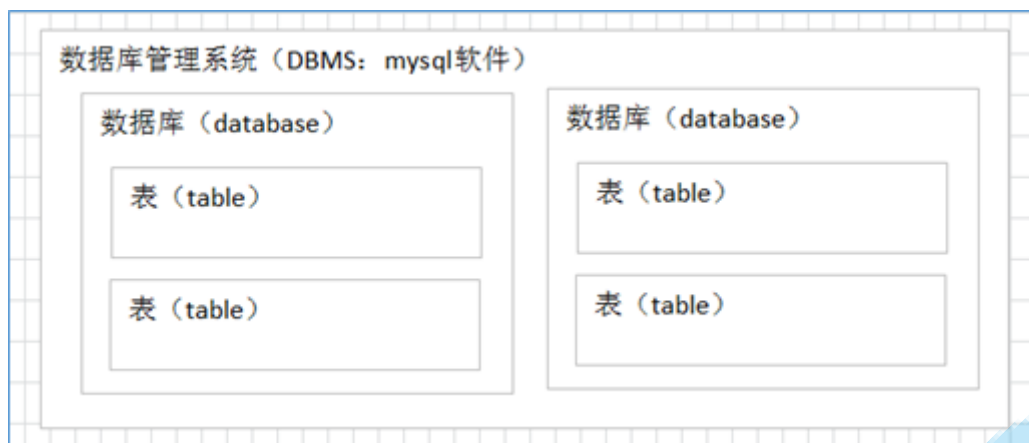
什么是数据库

数据库就是[存储数据的仓库]，其本质是一个[文件系统]，数据按照特定的格式将数据存储起来，用户可以通过SQL对数据库中的数据进行增加，修改，删除及查询操作。

什么是关系型数据库

数据库中的[记录是有行有列的数据库]就是关系型数据库，与之相反的就是NoSQL数据库了。

数据库和表



数据库管理系统（DataBase Management System, DBMS）：指一种[操作和管理数据库]的大型软件，用于建立、使用和维护数据库，对数据库进行统一管理和控制，以保证数据库的安全性和完整性。用户通过数据库管理系统访问数据库中表内的数据。（记录）

常见的数据库管理系统

- **MYSQL**：开源免费的数据库，小型的数据库。已经被Oracle收购了。MySQL5.5版本之后都是由Oracle发布的版本。
- **Oracle**：收费的大型数据库，Oracle公司的产品。Oracle收购SUN公司，收购MYSQL。
- **DB2**：IBM公司的数据库产品，收费的。常应用在银行系统中。在中国的互联网公司，要求去IOE（IBM小型机、Oracle数据库、EMC存储设备）
- **SQLServer**：Microsoft 公司收费的中型的数据库。C#、.net等语言常使用。
- **SyBase**：已经淡出历史舞台。提供了一个非常专业数据建模的工具PowerDesigner。
- **SQLite**：嵌入式的小型数据库，应用在手机端。

我们要学习的数据库：MySQL

MySQL介绍

MySQL是什么

MySQL 是最流行的【关系型数据库管理系统】，在WEB应用方面 MySQL是最好的RDBMS应用软件之一。

MySQL发展历程

- MySQL的历史可以追溯到1979年，一个名为Monty Widenius的程序员在为TcX的小公司打工，并且用BASIC设计了一个报表工具，使其可以在4MHz主频和16KB内存的计算机上运行。当时，这只是一个很底层的且仅面向报表的存储引擎，名叫Unireg。
- 1990年，TcX公司的客户中开始有人要求为他的API提供SQL支持。Monty直接借助于mSQL的代码，将它集成到自己的存储引擎中。令人失望的是，效果并不太令人满意，决心自己重写一个SQL支持。
- 1996年，MySQL 1.0发布，它只面向一小拨人，相当于内部发布。

- 到了1996年10月，MySQL 3.11.1发布(MySQL没有2.x版本)，最开始只提供Solaris下的二进制版本。一个月后，Linux版本出现了。在接下来的两年里，MySQL被依次移植到各个平台。
- 【1999~2000年】，【MySQL AB】公司在瑞典成立。Monty雇了几个人与Sleepycat合作，开发出了【Berkeley DB引擎】，由于BDB支持事务处理，因此MySQL从此开始支持事务处理了。
- 2000，MySQL不仅公布自己的源代码，并采用GPL(GNU General Public License)许可协议，正式进入开源世界。同年4月，MySQL对旧的存储引擎ISAM进行了整理，将其命名为MyISAM。
- 2001年，集成Heikki Tuuri的存储引擎【InnoDB】，这个引擎不仅能【支持事务处理，并且支持行级锁】。后来该引擎被证明是最为成功的MySQL事务存储引擎。【MySQL与InnoDB的正式结合版本是4.0】
- 2003年12月，【MySQL 5.0】版本发布，提供了视图、存储过程等功能。
- 【2008年1月】，【MySQL AB公司被Sun公司以10亿美金收购】，MySQL数据库进入Sun时代。在Sun时代，Sun公司对其进行了大量的推广、优化、Bug修复等工作。
- 2008年11月，MySQL 5.1发布，它提供了分区、事件管理，以及基于行的复制和基于磁盘的NDB集群系统，同时修复了大量的Bug。
- 【2009年4月】，Oracle公司以74亿美元收购Sun公司，自此MySQL数据库进入Oracle时代，而其第三方的存储引擎InnoDB早在2005年就被Oracle公司收购。
- 2010年12月，【MySQL 5.5发布】，其主要新特性包括半同步的复制及对SIGNAL/RESIGNAL的异常处理功能的支持，【最重要的是InnoDB存储引擎终于变为当前MySQL的默认存储引擎】。MySQL 5.5不是时隔两年后的一次简单的版本更新，而是加强了MySQL各个方面在企业级的特性。Oracle公司同时也承诺MySQL 5.5和未来版本仍是采用GPL授权的开源产品。

SQL介绍

什么是SQL

【SQL是Structured Query Language的缩写】，它的前身是著名的关系数据库原型系统System R所采用的SEQUEL语言。作为一种访问【关系型数据库的标准语言】，SQL自问世以来得到了广泛的应用，不仅是著名的大型商用数据库产品Oracle、DB2、Sybase、SQL Server支持它，很多开源的数据库产品如PostgreSQL、MySQL也支持它，甚至一些小型的产品如Access也支持SQL。近些年蓬勃发展的NoSQL系统最初是宣称不再需要SQL的，后来也不得不修正为Not Only SQL，来拥抱SQL。

蓝色巨人IBM对关系数据库以及SQL语言的形成和规范化产生了重大的影响，第一个版本的SQL标准SQL86就是基于System R的手册而来的。Oracle在1979年率先推出了支持SQL的商用产品。随着数据库技术和应用的发展，为不同RDBMS提供一致的语言成了一种现实需要。

对SQL标准影响最大的机构自然是那些著名的数据库产商，而具体的制订者则是一些非营利机构，例如【国际标准化组织ISO、美国国家标准委员会ANSI】等。各国通常会按照ISO标准和ANSI标准（这两个机构的很多标准是差不多等同的）制定自己的国家标准。中国是ISO标准委员会的成员国，也经常翻译一些国际标准对应的中文版。标准为了避免采用具体产品的术语，往往会抽象出很多名词，从而增加了阅读和理解难度，翻译成中文之后更容易词不达意。对于数据库系统实现者和用户而言，很多时候还不如直接读英文版本为好。虽然正式的标准不像RFC那样可以从网络上免费获得，标准草案还是比较容易找到的（例如：<http://www.jtc1sc32.org/doc/>）。待批准的标准草案和最终的标准也没有什么实质上的区别，能够满足日常工作的需要。

下面是SQL发展的简要历史：

1986年，ANSI X3.135-1986，ISO/IEC 9075:1986，SQL-86
1989年，ANSI X3.135-1989，ISO/IEC 9075:1989，SQL-89
1992年，ANSI X3.135-1992，ISO/IEC 9075:1992，SQL-92（SQL2）
1999年，ISO/IEC 9075:1999，SQL:1999（SQL3）

2003年, ISO/IEC 9075:2003, SQL:2003
2008年, ISO/IEC 9075:2008, SQL:2008
2011年, ISO/IEC 9075:2011, SQL:2011

如果要了解标准的内容, 比较推荐的方法是【泛读SQL92】(因为它涉及了SQL最基础和最核心的一些内容), 然后增量式的阅读其他标准。

不只是mysql还有其他数据库, 在SQL92或者SQL99这些国际SQL标准基础之上, 它们还扩展了自己的一些SQL语句, 比如MySQL中的limit关键字

SQL语言分类

- 数据定义语言: 简称【DDL】(Data Definition Language), 用来定义数据库对象: 数据库, 表, 列等。关键字: create, alter, drop等
- 数据操作语言: 简称【DML】(Data Manipulation Language), 用来对数据库中表的记录进行更新。关键字: insert, delete, update等
- 数据控制语言: 简称【DCL】(Data Control Language), 用来定义数据库的访问权限和安全级别, 及创建用户; 关键字: grant等
- 数据查询语言: 简称【DQL】(Data Query Language), 用来查询数据库中表的记录。关键字: select, from, where等

MySQL基础篇

MySQL单机安装(Linux)

操作系统: CentOS 7
MySQL: 5.6

MySQL的卸载

查看MySQL软件

```
rpm -qa|grep mysql  
yum repolist all | grep mysql
```

卸载MySQL

```
yum remove -y mysql mysql-libs mysql-common #卸载mysql  
rm -rf /var/lib/mysql #删除mysql下的数据文件  
rm /etc/my.cnf #删除mysql配置文件  
yum remove -y mysql-community-release-el6-5.noarch #删除组件
```

查看是否还有 MySQL 软件，有的话继续删除。

安装MySQL

```
#下载rpm文件
wget http://repo.mysql.com/mysql-community-release-el6-5.noarch.rpm
#执行rpm源文件
rpm -ivh mysql-community-release-el6-5.noarch.rpm
#执行安装文件
yum install mysql-community-server
```

启动MySQL

```
systemctl start mysqld
```

设置root用户密码

例如：为 root 账号设置密码为 root：

```
/usr/bin/mysqladmin -u root password 'root'
#没有密码 有原来的密码则加
/usr/bin/mysqladmin -u root -p '123' password 'root'
```

登录MySQL

- 登录命令

```
mysql -uroot -proot
```

- 命令说明：

-u: 指定数据库用户名
-p: 指定数据库密码，记住-u和登录密码之间没有空格

配置MySQL

```
vim /etc/my.cnf
```

修改内容如下：

```
[mysqld]
# MySQL设置大小写不敏感：默认：区分表名的大小写，不区分列名的大小写
# 0: 大小写敏感 1: 大小写不敏感
lower_case_table_names=1
# 默认字符集
character-set-server=utf8
```

MySQL远程连接授权

- 授权命令

```
grant 权限 on 数据库对象 to 用户
```

- 示例

授予root用户对所有数据库对象的全部操作权限:

```
mysql>GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'root' WITH GRANT OPTION;  
FLUSH PRIVILEGES;--刷新权限
```

- 命令说明:

- ALL PRIVILEGES :表示授予所有的权限, 此处可以指定具体的授权权限。
- *.* :表示所有库中的所有表
- 'root'@'%' : myuser是数据库的用户名, %表示是任意ip地址, 可以指定具体ip地址。
- IDENTIFIED BY 'mypassword' : mypassword是数据库的密码。

关闭linux的防火墙

```
systemctl stop firewalld (默认)  
systemctl disable firewalld.service (设置开启不启动)
```

客户端远程访问

利用navicat可以远程访问MySQL

注: 如果连接不上, 可以按以下步骤排错

1、MySQL是否正常启动

```
[root@localhost ~]# ps -ef | grep mysql  
root      1114      1  0 10:21 ?        00:00:00 /bin/sh /usr/bin/mysqld_safe -  
-datadir=/var/lib/mysql --socket=/var/lib/mysql/mysql.sock --pid-  
file=/var/run/mysqld/mysqld.pid --basedir=/usr --user=mysql  
mysql     1698     1114  0 10:21 ?        00:00:03 /usr/sbin/mysqld
```

2、查看防火墙是否关闭

```
[root@localhost ~]# systemctl status firewalld  
firewalld.service - firewalld - dynamic firewall daemon  
Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled)  
Active: inactive (dead)
```

3、查看root权限为所有ip都可以访问

```
mysql> show grants for root;  
+-----+  
+-----+  
| Grants for root@%  
  
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY PASSWORD  
'*81F5E21E35407D884A6CD4A731AEBFB6AF209E1B' WITH GRANT OPTION |  
+-----+
```

4、服务器与客户端是否可以ping通

```
ping 192.168.239.129
```

正在 Ping 192.168.239.129 具有 32 字节的数据:
来自 192.168.239.129 的回复: 字节=32 时间<1ms TTL=64
5、客户端是否可以telnet到服务器端
telnet 192.168.239.129 3306
6、Navicat是否正确安装

MySQL单机安装(Windows)

MySQL的卸载

- 1、打开控制面板上的卸载程序，选择相应的MySQL程序进行卸载

MySQL Connector J	Oracle Corporation
MySQL Connector Net 8.0.16	Oracle
MySQL Documents 5.7	Oracle Corporation
MySQL Examples and Samples 5.7	Oracle Corporation
MySQL Installer - Community	Oracle Corporation
MySQL Notifier 1.1.7	Oracle
MySQL Server 5.7	Oracle Corporation

- 2、删除MySQL文件夹，默认是C:\ProgramFiles\MySQL

- 3、删除MySQL数据文件目录和配置文件，默认是C:\ProgramData\MySQL

此电脑 > Windows (C:) > ProgramData > MySQL >		
名称	修改日期	类型
MySQL Server 5.7	2019/9/24 14:41	文件夹

安装MySQL

从MSQL官网下载MySQL服务器安装软件包，下载为版本“mysql-installer-community-5.7.26.0.msi”。

下载网址: <https://dev.mysql.com/downloads/mysql/>

Select Version:

5.7.27

Select Operating System:

Microsoft Windows

Select OS Version:

Windows (x86, 64-bit)

Looking for the latest GA version?

Recommended Download:

MySQL Installer
for Windows

All MySQL Products. For All Windows Platforms.
In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.

Windows (x86, 32 & 64-bit), MySQL Installer MSI

Go to Download Page >

Other Downloads:

<div>ZIP Archive</div> <div>(mysql-5.7.27-winx64.zip)</div>	5.7.27	321.5M	<div>MD5: c51bf575fd24b6c79ab25aa5465c6708 Signature</div> <div>Download</div>
<div>ZIP Archive, Debug Binaries & Test Suite</div> <div>(mysql-5.7.27-winx64-debug-test.zip)</div>	5.7.27	390.8M	<div>MD5: 7c63ac57351584d8065dc502283e29dd Signature</div> <div>Download</div>

!

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

双击安装包，出现安装选择对话框：

Choosing a Setup Type

Please select the Setup Type that suits your use case.

☒ **Developer Default**

Installs all products needed for MySQL development purposes.

☐ **Server only**

Installs only the MySQL Server product.

☐ **Client only**

Installs only the MySQL Client products, without a server.

☐ **Full**

Installs all included MySQL products and features.

☐ **Custom**

Manually select the products that should be installed on the system.

Setup Type Description

Installs the MySQL Server and the tools required for MySQL application development. This is useful if you intend to develop applications for an existing server.

This Setup Type includes:

* MySQL Server

* MySQL Shell

The new MySQL client application to manage MySQL Servers and InnoDB cluster instances.

* MySQL Router

High availability router daemon for InnoDB cluster setups to be installed on application nodes.

* MySQL Workbench

The GUI application to develop for and manage the server.

* MySQL for Excel

< Back







Next >

Cancel

点击"Next"后点击"Execute"

Installation

The following products will be installed.

Product	Status	Progress	Notes
 MySQL Server 5.7.26	Installing	88%	
 MySQL Notifier 1.1.7	Ready to Install		
 Connector/J 8.0.16	Ready to Install		
 Connector/NET 8.0.16	Ready to Install		
 MySQL Documentation 5.7.26	Ready to Install		
 Samples and Examples 5.7.26	Ready to Install		

[Show Details >](#)

< Back

Execute

Cancel

High Availability

☒ Standalone MySQL Server / Classic MySQL Replication

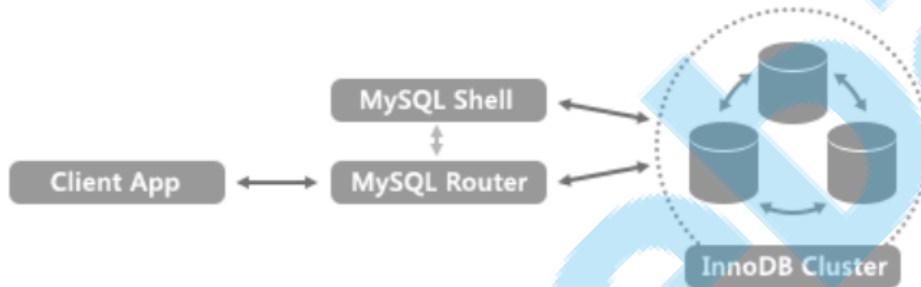
Choose this option to run the MySQL instance as a standalone database server with the opportunity to configure classic replication later. With this option, you can provide your own high-availability solution, if required.

☐ Sandbox InnoDB Cluster Setup (for testing only)

[Why is this option disabled?](#)

The [InnoDB cluster](#) technology provides an out-of-the-box HA (high availability) solution for MySQL using Group Replication.

This option enables you to test an InnoDB cluster setup on your local computer using several MySQL Server sandbox instances. Read about [Sandbox Deployment of InnoDB Cluster](#) to learn more.



To setup and configure a real-world production InnoDB cluster use the Create a New InnoDB Cluster or Add Local MySQL Server Instance to an InnoDB Cluster options available for MySQL Server 8.0.

InnoDB Cluster Log Verbosity Level

The log level for InnoDB cluster configuration.

MEDIUM

Next >

Cancel

Type and Networking

Server Configuration Type

Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.

Config Type: Development Computer

Connectivity

Use the following controls to select how you would like to connect to this server.

☒ TCP/IP Port: 3306

☒ Open Windows Firewall port for network access

☐ Named Pipe Pipe Name: MYSQL

☐ Shared Memory Memory Name: MYSQL

Advanced Configuration

Select the check box below to get additional configuration pages where you can set advanced and logging options for this server instance.

☐ Show Advanced and Logging Options

输入root账户的登录口令

Accounts and Roles

Root Account Password

Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password:

••••••

Repeat Password:

••••••

Password strength: **Weak**

MySQL User Accounts

Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role	
			<div><div>Add User</div><div>Edit User</div><div>Delete</div></div>

选择将MySQL作为windows服务启动

Windows Service

☒ Configure MySQL Server as a Windows Service

Windows Service Details

Please specify a Windows Service name to be used for this MySQL Server instance.
A unique name is required for each instance.

Windows Service Name:

☒ Start the MySQL Server at System Startup

Run Windows Service as ...

The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.

☒ Standard System Account

Recommended for most scenarios.

☐ Custom User

An existing user account can be selected for advanced scenarios.

执行即可

Apply Configuration

Click [Execute] to apply the changes

Configuration Steps

Log

- ☐ Writing configuration file
- ☐ Updating Windows Firewall rules
- ☐ Adjusting Windows service
- ☐ Initializing database (may take a long time)
- ☐ Starting the server
- ☐ Applying security settings
- ☐ Updating the Start menu link

< Back

Execute

Cancel

安装完后可以检测是否能够登录

Connect To Server

Here are the compatible MySQL Svr instances installed in this computer.
Please select the ones where the sample schemas and data will be created.


☐ Show MySQL Server instances maybe running in read-only mode

	Server	Port	Arch...	Type	Status
<input checked="" type="checkbox"/>	MySQL Server 5.7.26	3306	X64	Stand-alone Server	Connection succeeded.

Now give us the credentials we should use (needs to have root privileges).
Click "Check" to make sure they work.

User name: Credentials provided in Server configuration

Password:

 All connections succeeded.

Apply Configuration

The configuration operation has finished.

Configuration Steps

Log

- ✓ Checking if there are any features installed that need configuration.
- ✓ Running Scripts

The configuration for Samples and Examples 5.7.26 was successful.
Click Finish to continue.

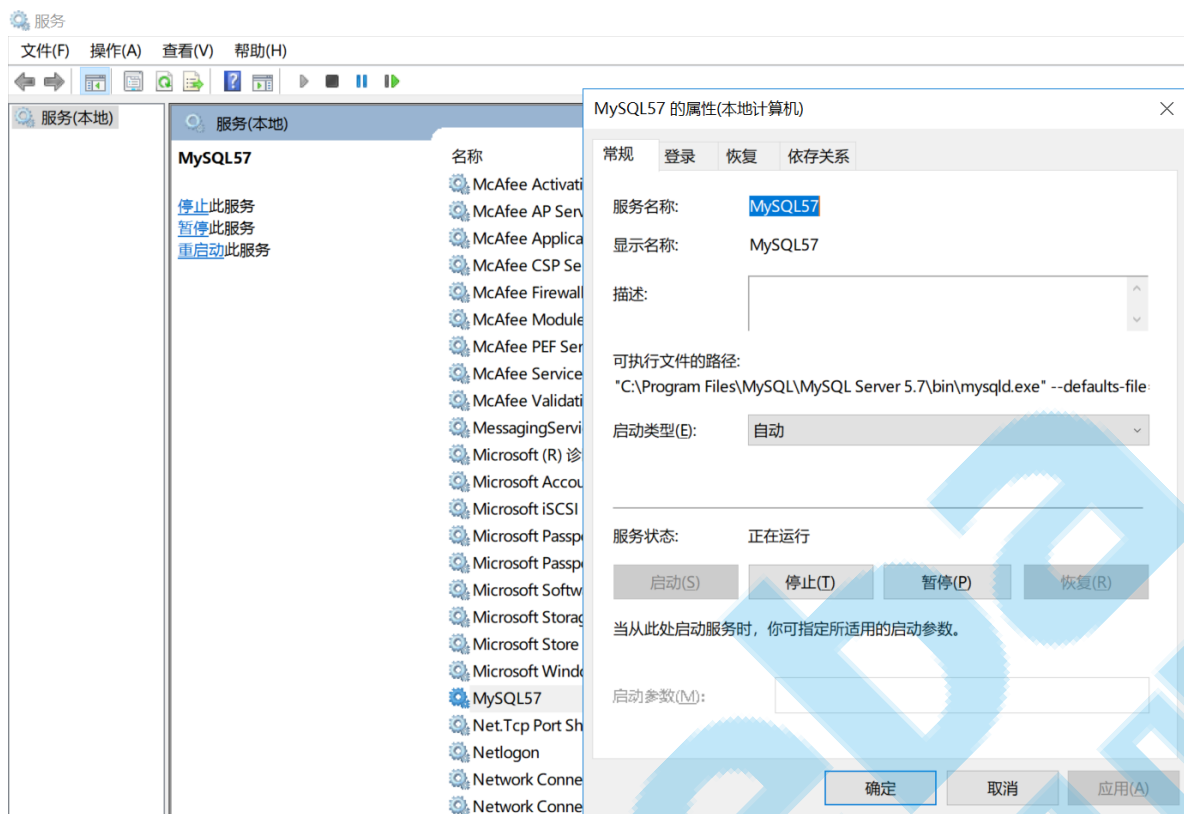
Finish

点击finish完成安装。(^_^)

启动MySQL

以windows服务的方式启动

MySQL安装的时候，如果是选择以Windows Service的方式启动，则在MySQL安装好后可以以windows服务的方式启动。打开windows本地服务，如下图：



此时MySQL是在windows启动后自动启动的服务，也可以选择启动类型为手动，这样可以节省机器性能。

常规

登录

恢复

依存关系

服务名称:

MySQL57

显示名称:

MySQL57

描述:

可执行文件的路径:

"C:\Program Files\MySQL\MySQL Server 5.7\bin\mysqld.exe" --defaults-file:

启动类型(E):

手动

服务状态:

已停止

启动(S)

停止(T)

暂停(P)

恢复(R)

当从此处启动服务时，你可指定所适用的启动参数。

启动参数(M):

确定

取消

应用(A)

点击启动按钮即可启动MySQL服务。

以应用程序的方式启动

找到mysql的安装目录

C:\Program Files\MySQL\MySQL Server 5.7\bin\				
名称	修改日期	类型	大小	
mysamchk.exe	2019/4/13 9:05	应用程序	3,814 KB	
mysamlog.exe	2019/4/13 9:05	应用程序	3,673 KB	
mysampack.exe	2019/4/13 9:05	应用程序	3,732 KB	
mysql.exe	2019/4/13 9:05	应用程序	4,102 KB	
mysql_config.pl	2019/4/13 16:33	PL 文件	8 KB	
mysql_config_editor.exe	2019/4/13 9:05	应用程序	3,803 KB	
mysql_embedded.exe	2019/4/13 9:05	应用程序	19,806 KB	
mysql_plugin.exe	2019/4/13 9:05	应用程序	3,483 KB	
mysql_secure_installation.exe	2019/4/13 9:05	应用程序	3,996 KB	
mysql_ssl_rsa_setup.exe	2019/4/13 9:05	应用程序	3,530 KB	
mysql_tzinfo_to_sql.exe	2019/4/13 9:05	应用程序	3,422 KB	
mysql_upgrade.exe	2019/4/13 9:05	应用程序	4,662 KB	
mysqladmin.exe	2019/4/13 9:05	应用程序	4,009 KB	
mysqlbinlog.exe	2019/4/13 9:05	应用程序	4,236 KB	
mysqlcheck.exe	2019/4/13 9:05	应用程序	4,032 KB	
mysqld.exe	2019/4/13 9:01	应用程序	38,716 KB	
mysqld_multi.pl	2019/4/13 16:33	PL 文件	28 KB	

双击mysqld即可使用。

注：该操作需要管理员权限。

登录MySQL

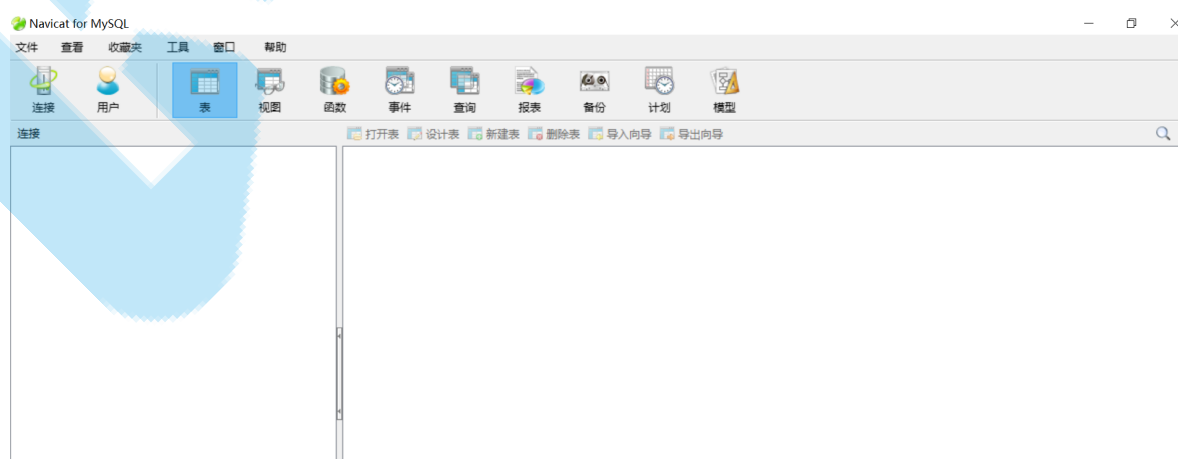
使用Navicat操作MySQL数据库

Navicat的安装

Navicat是一套快速、可靠并价格相当便宜的数据库管理工具，专为简化数据库的管理及降低系统管理成本而设。它的设计符合数据库管理员、开发人员及中小企业的需要。Navicat 是以直觉化的图形用户界面而建的，让你可以以安全并且简单的方式创建、组织、访问并共用信息。(百度百科)

使用navicat for mysql10.0.11简体中文绿色版解压后即可使用。(^_^)

打开navicat主界面



创建新连接

点击“连接”图标，出现创建新连接操作界面，如图：

填写：连接名、主机名或ip地址、端口、用户名、密码，并勾选保存密码

 新建连接 ✕

常规高级SSLSSHHTTP

连接名:

dbconn

主机名或 IP 地址:

localhost

端口:

3306

用户名:

root

密码:

●●●●●●

☒ 保存密码

连接测试

确定

取消

参数说明：

主机名：localhost或127.0.0.1（本机安装）

其他ip地址，如:192.168.12.128 (非本机安装，需要授权)

端口：3306 是mysql默认服务端口，不可以修改

用户名：root是MySQL的管理员账户拥有最高权限，一般开发都要有最高权限

密码：是root的账户密码，是MySQL在安装时设置的，一定不要忘记

点击连接测试，可以测试是否连接成功！

常规 高级 SSL SSH HTTP

连接名: dbconn

主机名或 IP 地址: localhost

端口: 3306

用户名: root

密码: ●●●●●●

连接成功

确定

连接测试 确定 取消

管理用户

新建用户

点击用户图标--新建用户

Navicat for MySQL

文件 查看 收藏夹 工具 窗口 帮助

连接 用户 表 视图 函数 事件 查询 报表 备份 计划 模型

连接

编辑用户 新建用户 删除用户 权限管理员

dbconn

- information_schema
- mysql
- performance_schema
- sakila
- sys
- world

mysql.session@localhost
mysql.sys@localhost
root@localhost

test1@localhost (dbconn) - 用户

文件	窗口	帮助
新建 保存		
常规	高级	服务器权限 权限 SQL 预览

用户名:

test1

主机:

localhost

密码:

●●●●●●●●

确认密码:

●●●●●●●●

可以赋权限

新建 保存

常规 高级 服务器权限 权限 SQL 预览

权限	授予
Select	<input checked="" type="checkbox"/>
Insert	<input checked="" type="checkbox"/>
Update	<input checked="" type="checkbox"/>
Delete	<input checked="" type="checkbox"/>
Create	<input checked="" type="checkbox"/>
Drop	<input checked="" type="checkbox"/>
Reload	<input checked="" type="checkbox"/>
Shutdown	<input checked="" type="checkbox"/>
Process	<input checked="" type="checkbox"/>
File	<input checked="" type="checkbox"/>
Grant Option	<input checked="" type="checkbox"/>
References	<input checked="" type="checkbox"/>
Index	<input checked="" type="checkbox"/>
Alter	<input checked="" type="checkbox"/>
Show Databases	<input checked="" type="checkbox"/>
Super	<input checked="" type="checkbox"/>

修改用户

编辑用户 新建用户 删除用户 权限管理员

test1@localhost (dbconn-test1) - 用户

文件 窗口 帮助

新建 保存

常规 高级 服务器权限 权限 SQL 预览

用户名: test1

主机: localhost

密码: ●●●●●●●●

确认密码: ●●●●●●●●

删除用户

编辑用户 新建用户 删除用户 权限管理员

mysql.session@localhost

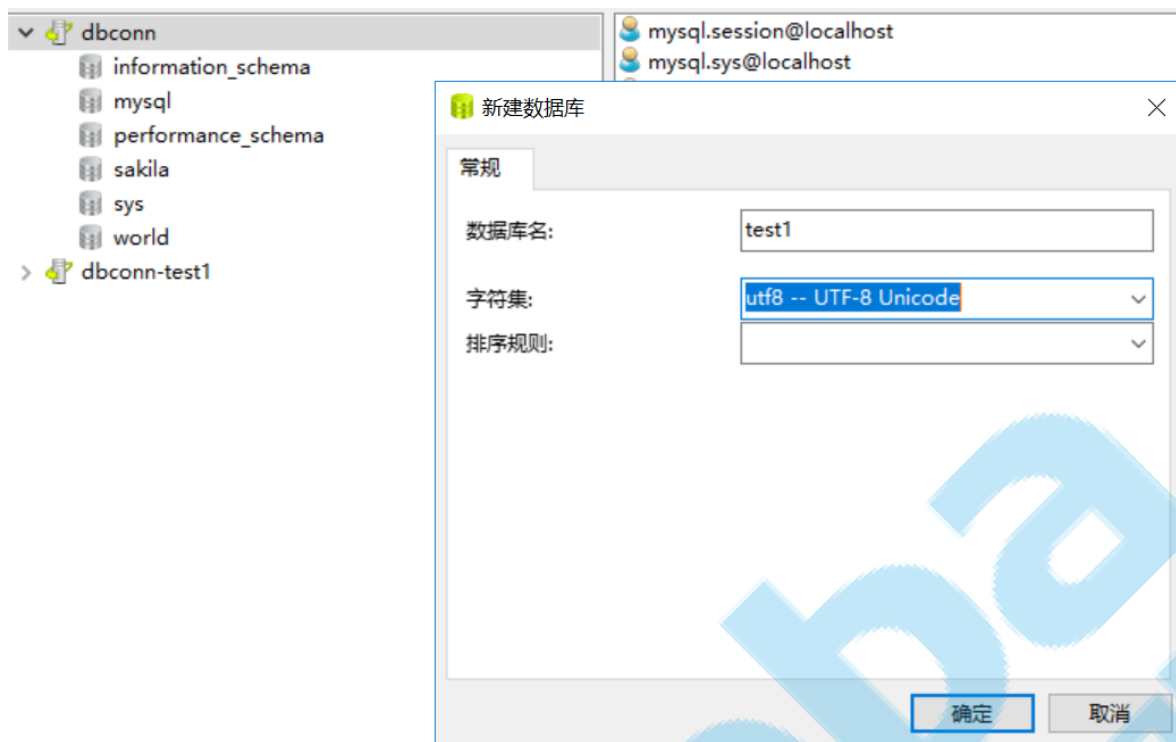
mysql.sys@localhost

root@localhost

test1@localhost

创建数据库

在数据库上点击右键，选择新建数据库，填写数据库名和字符集，字符集选择utf8



创建表

打开数据库，点击"新建表"



主键 可以选择自动递增。

表数据的基本操作

在表中直接操作（不推荐）

tuser @test1 (dbconn) - 表

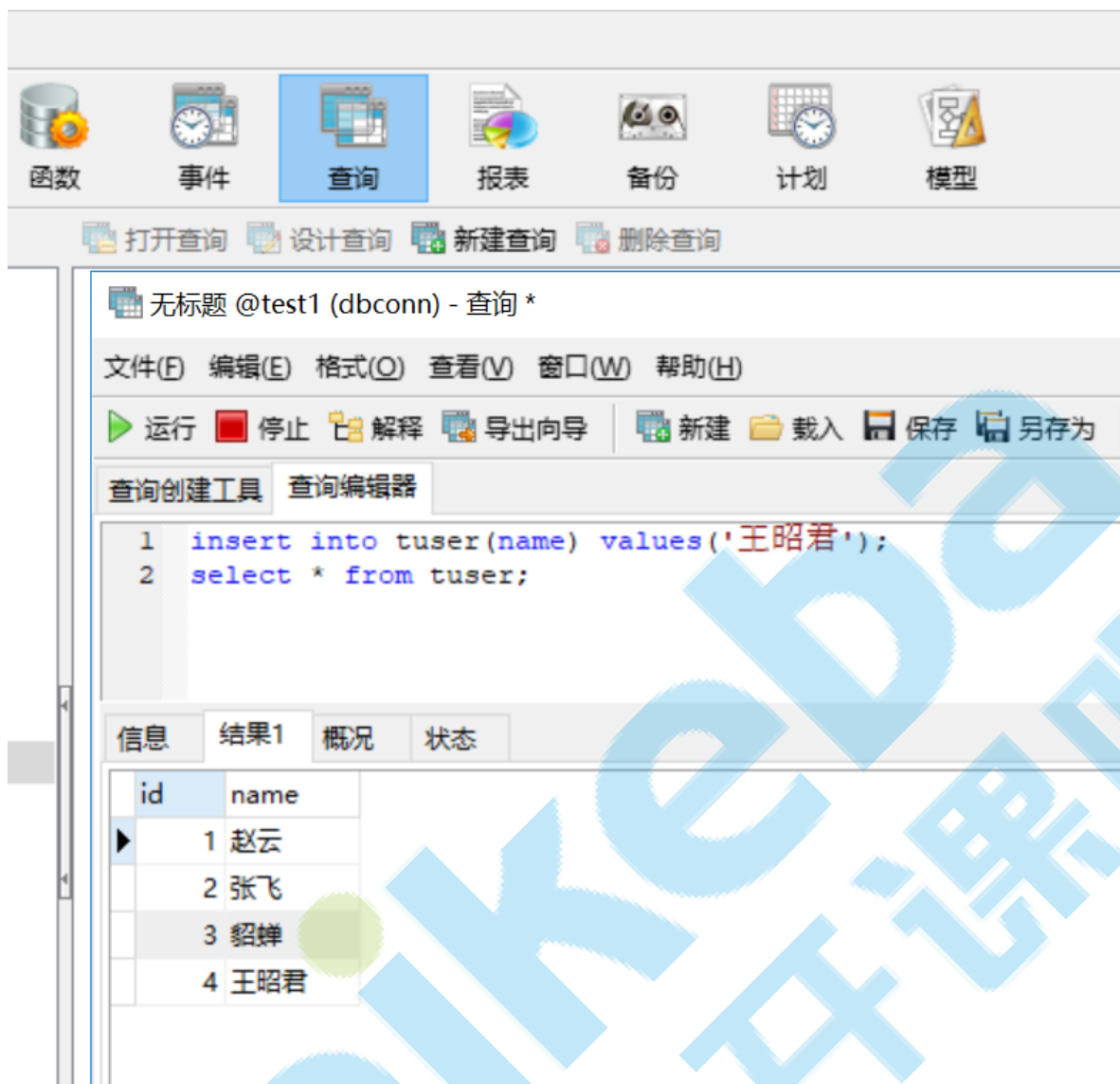
文件 编辑 查看 窗口 帮助	
导入向导	导出向导
筛选向导	网格查看
表单查看	备注
十六进制	图像
A-z	
id	name
1	赵云
2	张飞
3	貂蝉



使用SQL语句操作

点击“查询”，点击“新建查询”，打开SQL窗口就可以编写SQL语句了。

点击“运行”，可以运行编写的SQL语句。



DDL语句

数据库操作：database

创建数据库

```
create database 数据库名;
create database 数据库名 character set 字符集;
```

查看数据库

查看数据库服务器中的所有数据库:

```
show databases;
```

查看某个数据库的定义的信息:

```
show create database 数据库名;
```

删除数据库（慎用）

```
drop database 数据库名称;
```

其他数据库操作命令

切换数据库：

```
use 数据库名;
```

查看正在使用的数据库：

```
select database();
```

表操作：table

字段类型

- 常用的类型有：

数字型：int

浮点型：double

字符型：varchar（可变长字符串）

日期类型：date（只有年月日，没有时分秒）

datetime（年月日，时分秒）

boolean类型：不支持，一般使用tinyint替代（值为0和1）

分类	类型名称	说明
整数类型	tinyInt	很小的整数
	smallint	小的整数
	mediumint	中等大小的整数
	int(integer)	普通大小的整数
小数类型	float	单精度浮点数
	double	双精度浮点数
	decimal(m,d)	压缩严格的定点数-----开发时用
日期类型	year	YYYY 1901~2155
	time	HH:MM:SS -838:59:59~838:59:59
	date	YYYY-MM-DD 1000-01-01~9999-12-3
	datetime-开发用	YYYY-MM-DD HH:MM:SS 1000-01-01 00:00:00~ 9999-12-31 23:59:59
	timestamp	YYYY-MM-DD HH:MM:SS 1970~01~01 00:00:01 UTC~2038-01-19 03:14:07UTC

文本、二进制类型	CHAR(M)	M 为 0~255 之间的整数
	VARCHAR(M)	M 为 0~65535 之间的整数
	TINYBLOB	允许长度 0~255 字节
	BLOB	允许长度 0~65535 字节
	MEDIUMBLOB	允许长度 0~167772150 字节
	LOBLOB	允许长度 0~4294967295 字节
	TINYTEXT	允许长度 0~255 字节

创建表

```
create table 表名(  
    字段名 类型(长度) 约束,  
    字段名 类型(长度) 约束  
);
```

单表约束:

- 主键约束: `primary key`
- 唯一约束: `unique`
- 非空约束: `not null`

注意:

主键约束 = 唯一约束 + 非空约束

查看表

查看数据库中的所有表:

```
show tables;
```

查看表结构:

```
desc 表名;
```

删除表

```
drop table 表名;
```

修改表

```
alter table 表名 add 列名 类型(长度) 约束;      --修改表添加列.

alter table 表名 modify 列名 类型(长度) 约束;    --修改表修改列的类型长度及约束.

alter table 表名 change 旧列名 新列名 类型(长度) 约束;  --修改表修改列名.

alter table 表名 drop 列名;                      --修改表删除列.

rename table 表名 to 新表名;                    --修改表名

alter table 表名 character set 字符集;          --修改表的字符集
```

DML语句

插入记录: insert

- 语法:

```
insert into 表 (列名1,列名2,列名3..) values (值1,值2,值3..); -- 向表中插入某些列

insert into 表 values (值1,值2,值3..); --向表中插入所有列

insert into 表 (列名1,列名2,列名3..) select (列名1,列名2,列名3..) from 表

insert into 表 values select * from 表
```

- 注意:
 1. 列名数与values后面的值的个数相等
 2. 列的顺序与插入的值得顺序一致
 3. 列名的类型与插入的值要一致.
 4. 插入值得时候不能超过最大长度.
 5. 值如果是字符串或者日期需要加引号" (一般是单引号)
- 例如:

```
INSERT INTO sort(sid,sname) VALUES('s001', '电器');

INSERT INTO sort(sid,sname) VALUES('s002', '服饰');

INSERT INTO sort VALUES('s003', '化妆品');

INSERT INTO sort VALUES('s004', '书籍');
```

更新记录: update

- 语法:

```
update 表名 set 字段名=值,字段名=值;

update 表名 set 字段名=值,字段名=值 where 条件;
```

- 注意:

1. 列名的类型与修改的值要一致.
2. 修改值得时候不能超过最大长度.
3. 值如果是字符串或者日期需要加“.”.

删除记录: delete

- 语法:

```
delete from 表名 [where 条件];
```

- 面试题:

删除表中所有记录使用【delete from 表名】, 还是用【truncate table 表名】?

删除方式:

- delete : 一条一条删除, 不清空auto_increment记录数.
- truncate : 直接将表删除, 重新建表, auto_increment将置为零, 从新开始.

DQL语句

准备工作

创建商品表:

案例演示:

#商品表

```
CREATE TABLE product (
```

```
    pid INT PRIMARY KEY AUTO_INCREMENT, # 自增加 AUTO_INCREMENT
```

```
    pname VARCHAR(20), #商品名称
```

```
    price DOUBLE, #商品价格
```

```
    pdate DATE, # 日期
```

```
    cid int #分类ID
```

```
);
```

#目录表

```
create table category(
```

```
    id INT PRIMARY KEY ,
```

```
    cname varchar(100)
```

```
);
```

```
INSERT INTO product VALUES(NULL, '泰国大榴莲', 98, NULL, 1);
```

```
INSERT INTO product VALUES(NULL, '泰国大枣', 38, NULL, 1);
```

```
INSERT INTO product VALUES(NULL, '新疆切糕', 68, NULL, 2);
```

```
INSERT INTO product VALUES(NULL, '十三香', 10, NULL, 2);
```

```
INSERT INTO product VALUES(NULL, '泰国大枣', 20, NULL, 2);
```

```
insert into product values(null, '泰国大枣', 98, null, 20); #没有对应
```

```
insert into product values(null, 'iPhone手机', 800, null, 30); #没有对应
```

```
INSERT INTO category VALUES(1, '国外食品');
```

```
INSERT INTO category VALUES(2, '国内食品');
```

```
INSERT INTO category VALUES(3, '国内服装'); #没有对应
```

完整DQL语法顺序:

```
SELECT DISTINCT
    < select_list >
FROM
    < left_table > < join_type >
JOIN < right_table > ON < join_condition >
WHERE
    < where_condition >
GROUP BY
    < group_by_list >
HAVING
    < having_condition >
ORDER BY
    < order_by_condition >
LIMIT < limit_number >
```

简单查询

• SQL语法关键字:

```
SELECT
FROM
```

• 案例:

1. 查询所有的商品.

```
select * from product;
```

1. 查询商品名和商品价格.

```
select pname,price from product;
```

1. 别名查询, 使用的as关键字, as可以省略的.

表别名:

```
select * from product as p;
```

列别名:

```
select pname as pn from product;
```

1. 去掉重复值.

```
select distinct price from product;
```

1. 查询结果是表达式（运算查询）：将所有商品的价格+10元进行显示.

```
select pname,price+10 from product;
```

条件查询

- SQL语法关键字:

WHERE

- 案例:

1. 查询商品名称为十三香的商品所有信息:

```
select * from product where pname = '十三香';
```

1. 查询商品价格>60元的所有的商品信息:

```
select * from product where price > 60;
```

- where后的条件写法:

> ,< ,= ,>= ,<= ,<>

like 使用占位符 _ 和 % _代表一个字符 %代表任意个字符.

```
select * from product where pname like '%新%';
```

in在某个范围中获得值(exists).

```
select * from product where pid in (2,5,8);
```

比较运算符	> < <= >= = <>	大于、小于、大于(小于)等于、不等于
	BETWEEN ...AND...	显示在某一区间的值(含头含尾)
	IN(set)	显示在 in 列表中的值，例：in(100,200)
	LIKE '张_'	模糊查询，Like 语句中，% 代表零个或多个任意字符，_ 代表一个字符，例 first_name like '_a%' ;
	IS NULL	判断是否为空
逻辑运算符	and	多个条件同时成立
	or	多个条件任一成立
	not	不成立，例：where not(salary>100);

排序

- SQL语法关键字:

ORDER BY

ASC (升序) DESC (降序)

- 案例:

- 查询所有的商品，按价格进行排序.(asc-升序,desc-降序)

```
select * from product order by price;
```

- 查询名称有新的商品的信息并且按价格降序排序.

```
select * from product where pname like '%新%' order by price desc;
```

聚合函数 (组函数)

- 特点：只对单列进行操作
- 常用的聚合函数：

`sum()`: 求某一列的和

`avg()`: 求某一列的平均值

`max()`: 求某一列的最大值

`min()`: 求某一列的最小值

`count()`: 求某一列的元素个数

- **案例:**

1. 获得所有商品的价格的总和:

```
select sum(price) from product;
```

1. 获得所有商品的平均价格:

```
select avg(price) from product;
```

1. 获得所有商品的个数:

```
select count(*) from product;
```

分组

- **SQL语法关键字:**

`GROUP BY`

`HAVING`

- **案例:**

1. 根据cno字段分组, 分组后统计商品的个数.

```
select cid,count(*) from product group by cid;
```

1. 根据cno分组, 分组统计每组商品的平均价格, 并且平均价格> 60;

```
select cid,avg(price) from product group by cid having avg(price)>60;
```

- **注意事项:**

1. `select`语句中的列（非聚合函数列），必须出现在`group by`子句中
1. `group by`子句中的列，不一定要出现在`select`语句中
1. 聚合函数只能出现`select`语句中或者`having`语句中，一定不能出现在`where`语句中。

分页查询

关键字：

```
limit [offset,] rows
```

`limit` 关键字不是 SQL92 标准提出的关键字，它是 MySQL 独有的语法。

通过 `limit` 关键字，MySQL 实现了物理分页。

分页分为逻辑分页和物理分页：

逻辑分页：将数据库中的数据查询到内存之后再进行分页。

物理分页：通过 `limit` 关键字，直接在数据库中进行分页，最终返回的数据，只是分页后的数据。

- 格式：

```
SELECT * FROM table LIMIT [offset,] rows
```

`offset`：偏移量

`rows`：每页多少行记录。

- 案例

分页查询商品表，每页3条记录，查第一页

子查询

- 定义

子查询允许把一个查询嵌套在另一个查询当中。

子查询，又叫内部查询，相对于内部查询，包含内部查询的就称为外部查询。

子查询可以包含普通select可以包括的任何子句，比如：distinct、group by、order by、limit、join和union等；

但是对应的外部查询必须是以下语句之一：select、insert、update、delete。

- 位置

select中、from 后、where 中。

group by 和order by 中无实用意义。

- 举例

查询“化妆品”分类下的商品信息

其他查询语句

`union` 集合的并集（不包含重复记录）

`unionall` 集合的并集（包含重复记录）

SQL解析顺序

接下来再走一步，让我们看看一条SQL语句的前世今生。

首先看一下示例语句：

```
SELECT DISTINCT
    < select_list >
FROM
    < left_table > < join_type >
JOIN < right_table > ON < join_condition >
WHERE
    < where_condition >
GROUP BY
    < group_by_list >
HAVING
    < having_condition >
ORDER BY
    < order_by_condition >
LIMIT < limit_number >
```

然而它的执行顺序是这样的：

```

-- 行过滤
1 FROM <left_table>
2 ON <join_condition>
3 <join_type> JOIN <right_table>    第二步和第三步会循环执行
4 WHERE <where_condition>          第四步会循环执行，多个条件的执行顺序是从左往右的。
5 GROUP BY <group_by_list>
6 HAVING <having_condition>
-- 列过滤
7 SELECT                            分组之后才会执行SELECT
8 DISTINCT <select_list>
-- 排序
9 ORDER BY <order_by_condition>
-- MySQL附加
10 LIMIT <limit_number>             前9步都是SQL92标准语法。limit是MySQL的独有语法。

```

虽然自己没想到是这样的，不过一看还是很自然和谐的，从哪里获取，不断的过滤条件，要选择一样或不一样的，排好序，那才知道要取前几条呢。

既然如此了，那就让我们根据案例一步步来看看其中的细节吧。

现在开始SQL解析之旅吧！

1.FROM

对FROM的左边的表和右边的表计算笛卡尔积(CROSS JOIN)。产生虚表VT1

```
mysql> select * from product,category;
```

pid	pname	price	pdate	cid	id	cname
1	泰国大榴莲	98	NULL	1	1	国外食品
1	泰国大榴莲	98	NULL	1	2	国内食品
1	泰国大榴莲	98	NULL	1	3	国内服装
2	泰国大枣	38	NULL	1	1	国外食品
2	泰国大枣	38	NULL	1	2	国内食品
2	泰国大枣	38	NULL	1	3	国内服装
3	新疆切糕	68	NULL	2	1	国外食品
3	新疆切糕	68	NULL	2	2	国内食品
3	新疆切糕	68	NULL	2	3	国内服装
4	十三香	10	NULL	2	1	国外食品
4	十三香	10	NULL	2	2	国内食品
4	十三香	10	NULL	2	3	国内服装
5	泰国大枣	20	NULL	2	1	国外食品
5	泰国大枣	20	NULL	2	2	国内食品
5	泰国大枣	20	NULL	2	3	国内服装
6	泰国大枣	98	NULL	20	1	国外食品
6	泰国大枣	98	NULL	20	2	国内食品
6	泰国大枣	98	NULL	20	3	国内服装
7	iPhone手机	800	NULL	30	1	国外食品
7	iPhone手机	800	NULL	30	2	国内食品
7	iPhone手机	800	NULL	30	3	国内服装

```

21 rows in set (0.00 sec)

```


2.ON过滤

对 虚表VT1 进行ON筛选，只有那些符合的行才会被记录在虚表VT2中。

注意：这里因为语法限制，使用了'WHERE'代替，从中读者也可以感受到两者之间微妙的关系；

```
mysql> select * from product a , category b where a.cid=b.id;
```

pid	pname	price	pdate	cid	id	cname
1	泰国大榴莲	98	NULL	1	1	国外食品
2	泰国大枣	38	NULL	1	1	国外食品
3	新疆切糕	68	NULL	2	2	国内食品
4	十三香	10	NULL	2	2	国内食品
5	泰国大枣	20	NULL	2	2	国内食品

5 rows in set (0.00 sec)

3.OUTER JOIN添加外部行

如果指定了OUTER JOIN（比如left join、right join），那么 保留表中未匹配的行 就会作为外部行 添加到 虚拟表VT2 中，产生 虚拟表VT3。

如果FROM子句中包含两个以上的表的话，那么就会对上一个join连接产生的结果VT3和下一个表重复执行步骤1~3这三个步骤，一直到处理完所有的表为止。

```
mysql> select * from product a left outer join category b on a.cid=b.id; # 以左表数据为准
```

pid	pname	price	pdate	cid	id	cname
1	泰国大榴莲	98	NULL	1	1	国外食品
2	泰国大枣	38	NULL	1	1	国外食品
3	新疆切糕	68	NULL	2	2	国内食品
4	十三香	10	NULL	2	2	国内食品
5	泰国大枣	20	NULL	2	2	国内食品
6	泰国大枣	98	NULL	20	NULL	NULL
7	iPhone手机	800	NULL	30	NULL	NULL

7 rows in set (0.00 sec)

```
mysql> select * from product a right outer join category b on a.cid=b.id; #以右表数据为准
```

pid	pname	price	pdate	cid	id	cname
1	泰国大榴莲	98	NULL	1	1	国外食品
2	泰国大枣	38	NULL	1	1	国外食品
3	新疆切糕	68	NULL	2	2	国内食品
4	十三香	10	NULL	2	2	国内食品
5	泰国大枣	20	NULL	2	2	国内食品
NULL	NULL	NULL	NULL	NULL	3	国内服装

6 rows in set (0.00 sec)

4.WHERE

对 虚拟表VT3 进行WHERE条件过滤。只有符合的记录才会被插入到 虚拟表VT4 中。

注意:

此时因为分组, 不能使用聚合运算; 也不能使用SELECT中创建的别名;

与ON的区别:

- 如果有外部列, ON针对过滤的是关联表, 主表(保留表)会返回所有的列;
- 如果没有添加外部列, 两者的效果是一样的;

应用:

- 对主表的过滤应该放在WHERE;
- 对于关联表, 先条件查询后连接则用ON, 先连接后条件查询则用WHERE;

```
mysql> select * from product a left outer join category b on a.cid=b.id where a.pname='泰国大枣';
```

pid	pname	price	pdate	cid	id	cname
2	泰国大枣	38	NULL	1	1	国外食品
5	泰国大枣	20	NULL	2	2	国内食品
6	泰国大枣	98	NULL	20	NULL	NULL

3 rows in set (0.00 sec)

5.GROUP BY

根据group by子句中的列, 对VT4中的记录进行分组操作, 产生 虚拟表VT5。

注意:

其后处理过程的语句, 如SELECT,HAVING, 所用到的列必须包含在GROUP BY中。对于没有出现的, 得用聚合函数;

原因:

GROUP BY改变了对表的引用, 将其转换为新的引用方式, 能够对其进行下一级逻辑操作的列会减少;

我的理解是:

根据分组字段, 将具有相同分组字段的记录归并成一条记录, 因为每一个分组只能返回一条记录, 除非是被过滤掉了, 而不在分组字段里面的字段可能会有多个值, 多个值是无法放进一条记录的, 所以必须通过聚合函数将这些具有多值的列转换成单值;

```
mysql> select * from product a left outer join category b on a.cid=b.id where a.pname='泰国大枣' group by a.price;
```

pid	pname	price	pdate	cid	id	cname
5	泰国大枣	20	NULL	2	2	国内食品
2	泰国大枣	38	NULL	1	1	国外食品
6	泰国大枣	98	NULL	20	NULL	NULL

3 rows in set (0.01 sec)

6.HAVING

对 虚拟表VT5 应用having过滤，只有符合的记录才会被 插入到 虚拟表VT6 中。

```
mysql> select * from product a left outer join category b on a.cid=b.id where a.pname='泰国大枣' group by a.price having b.id <=2;
```

pid	pname	price	pdate	cid	id	cname
5	泰国大枣	20	NULL	2	2	国内食品
2	泰国大枣	38	NULL	1	1	国外食品

2 rows in set (0.00 sec)

7.SELECT

这个子句对SELECT子句中的元素进行处理，生成VT5表。

(5-J1)计算表达式 计算SELECT 子句中的表达式，生成VT5-J1

8.DISTINCT

寻找VT5-1中的重复列，并删掉，生成VT5-J2

如果在查询中指定了DISTINCT子句，则会创建一张内存临时表（如果内存放不下，就需要存放在硬盘了）。这张临时表的表结构和上一步产生的虚拟表VT5是一样的，不同的是对进行DISTINCT操作的列增加了一个唯一索引，以此来除重复数据。

```
mysql> select distinct a.pname from product a left outer join category b on a.cid=b.id where a.pname='泰国大枣' group by a.price ;
```

pname
泰国大枣

1 row in set (0.00 sec)

9.ORDER BY

从 VT5-J2 中的表中，根据ORDER BY 子句的条件对结果进行排序，生成VT6表。

注意：

唯一可使用SELECT中别名的地方；

```
mysql> select * from product a left outer join category b on a.cid=b.id where
a.pname='泰国大枣' group by a.price having b.id <=2 order by b.id;
+-----+-----+-----+-----+-----+-----+-----+
| pid | pname      | price | pdate | cid | id  | cname      |
+-----+-----+-----+-----+-----+-----+-----+
| 2   | 泰国大枣   | 38    | NULL  | 1   | 1   | 国外食品   |
| 5   | 泰国大枣   | 20    | NULL  | 2   | 2   | 国内食品   |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

10.LIMIT (MySQL特有)

LIMIT子句从上一步得到的VT6虚拟表 中选出从指定位置开始的指定行数据。

注意：

offset 和 rows 的正负带来的影响；

当偏移量很大时效率是很低的，可以这么做：

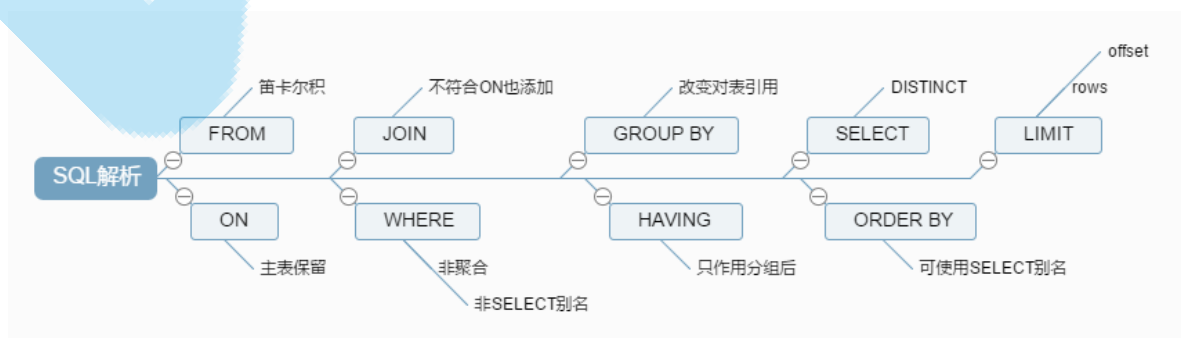
采用子查询的方式优化，在子查询里先从索引获取到最大id，然后倒序排，再取N行结果集

采用INNER JOIN优化，JOIN子句里也优先从索引获取ID列表，然后直接关联查询获得最终结果

```
mysql> select * from product a left outer join category b on a.cid=b.id where
a.pname='泰国大枣' group by a.price having b.id <=2 order by b.id limit 1;
+-----+-----+-----+-----+-----+-----+-----+
| pid | pname      | price | pdate | cid | id  | cname      |
+-----+-----+-----+-----+-----+-----+-----+
| 2   | 泰国大枣   | 38    | NULL  | 1   | 1   | 国外食品   |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

解析顺序总结

图示



流程分析

1. FROM (将最近的两张表, 进行笛卡尔积) ---VT1
2. ON (将VT1按照它的条件进行过滤) ---VT2
3. LEFT JOIN (保留左表的记录) ---VT3
4. WHERE (过滤VT3中的记录) --VT4...VTn
5. GROUP BY (对VT4的记录进行分组) ---VT5
6. HAVING (对VT5中的记录进行过滤) ---VT6
7. SELECT (对VT6中的记录, 选取指定的列) --VT7
8. ORDER BY (对VT7的记录进行排序) --VT8
9. LIMIT (对排序之后的值进行分页) --MySQL特有的语法

流程说明:

- **单表查询:** 根据 WHERE 条件过滤表中的记录, 形成中间表 (这个中间表对用户是不可见的); 然后根据 SELECT 的选择列选择相应的列进行返回最终结果。
- **两表连接查询:** 对两表求积 (笛卡尔积) 并用 ON 条件和连接连接类型进行过滤形成中间表; 然后根据 WHERE 条件过滤中间表的记录, 并根据 SELECT 指定的列返回查询结果。

笛卡尔积: 行相乘、列相加。

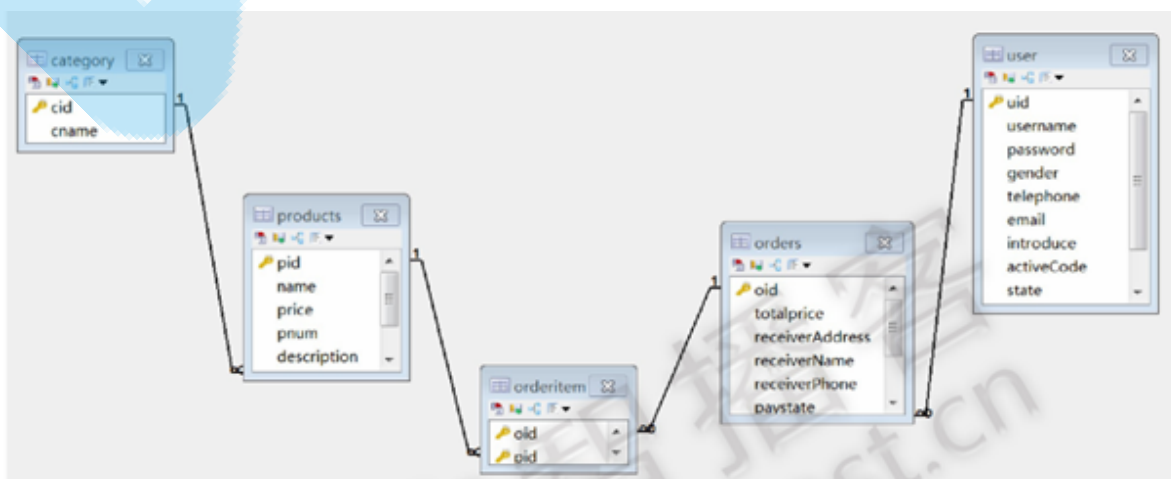
- **多表连接查询:** 先对第一个和第二个表按照两表连接做查询, 然后用查询结果和第三个表做连接查询, 以此类推, 直到所有的表都连接上为止, 最终形成一个中间的结果表, 然后根据 WHERE 条件过滤中间表的记录, 并根据 SELECT 指定的列返回查询结果。

WHERE条件解析顺序

1. MySQL: 从左往右去执行 WHERE 条件的。
2. Oracle: 从右往左去执行 WHERE 条件的。

写 WHERE 条件的时候, 优先级高的部分要去编写过滤力度最大的条件语句。

多表之间的关系



如上图所示，实际业务数据库中的表之间都是有关系的，我们接下来主要要学习的就是如何分析表关系及建立表关系。

1. 分类表

```
create table category(  
    cid varchar(32) primary key,  
    cname varchar(100)  
);
```

1. 商品表

```
create table product(  
    pid varchar(32) primary key,  
    pname varchar(40),  
    price double  
);
```

1. 订单表

```
create table orders(  
    oid varchar(32) primary key,  
    totalprice double  
);
```

1. 订单项表

```
create table orderitem(  
    oid varchar(50),  
    pid varchar(50)  
);
```

表与表之间的关系

表与表之间的关系，说的就是表与表之间数据的关系。

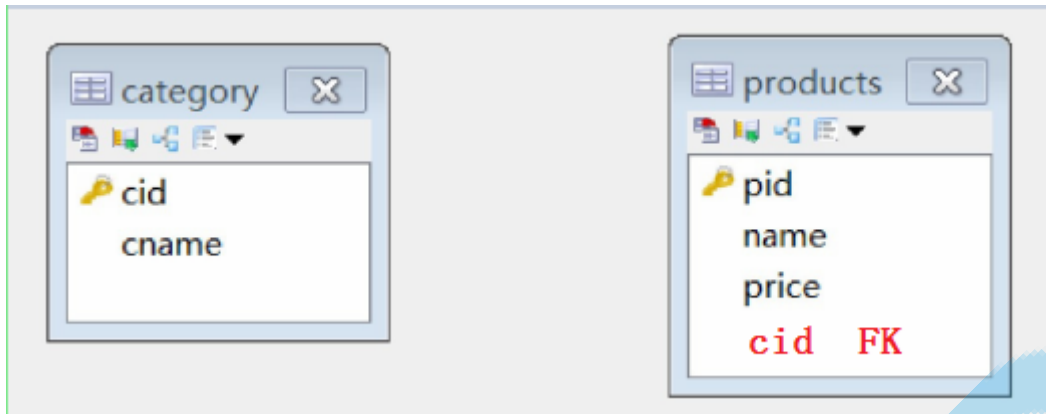
- **一对一关系**
常见实例：一夫一妻
- **一对多关系**
常见实例：会员和订单
- **多对多关系（需要中间表实现）**
常见实例：商品和订单

外键

如何表示表与表之间的关系呢？就是使用**外键约束**表示的。

要想理解外键，我们先去理解表的角色：**主表和从表**（需要建立关系才有了主从表的角色区分）

- 主从表的理解



现在我们有两张表“分类表”和“商品表”。

目前从表的声明上来说，没有关系，但是我们有个需求：

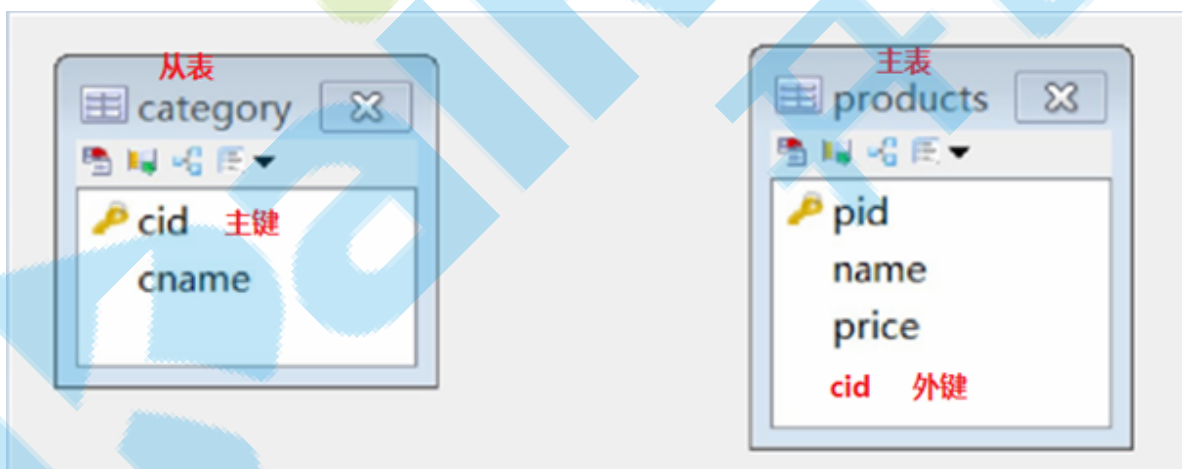
商品应该有所属的分类，这个时候需要将分类表和商品表建立关系，如何建立？

按照以上需求分析：

主表是：商品表。主表中，应该有一个字段去关联从表，而这个关联字段就是外键。

从表是：分类表。从表中，应该有一个字段去关联主表，而这个关联字段就是主键。

- 主键外键的理解



- 如何操作外键

主表添加外键的格式：

```
alter table 表名 add [constraint] [约束名称] foreign key (主表外键字段) references 从表(从表主键)
```

主表删除外键的格式：

```
alter table 表名 drop foreign key 外键约束名称
```

使用外键目的：

保证数据完整性（数据保存在多张表中的时候）

在互联网项目中，一般情况下，不建议建立外键关系。

一对一关系（了解）

在实际工作中，一对一在开发中应用不多，因为一对一完全可以创建成一张表

案例：一个丈夫只能有一个妻子

- 建表语句：

```
CREATE TABLE wife(  
    id INT PRIMARY KEY ,  
    wname VARCHAR(20),  
    sex CHAR(1)  
);  
  
CREATE TABLE husband(  
    id INT PRIMARY KEY ,  
    hname VARCHAR(20),  
    sex CHAR(1)  
);
```

- 一对一关系创建方式1之**外键唯一**：
 - 添加外键列wid，指定该列的约束为唯一（不加唯一约束就是一对多关系）

```
ALTER TABLE husband ADD wid INT UNIQUE;
```

- 添加外键约束

```
alter table husband add foreign key (wid) references wife(id);
```

- 一对一关系创建方式2之**主键做外键**：（课后作业）
 - 思路：使用主表的主键作为外键去关联从表的主键

一对多关系

案例：一个分类对应多个商品

总结：

有外键的就是多的一方。

注意事项：

一对多关系和一对一关系的创建很类似，唯一区别就是外键不唯一。

一对多关系创建：

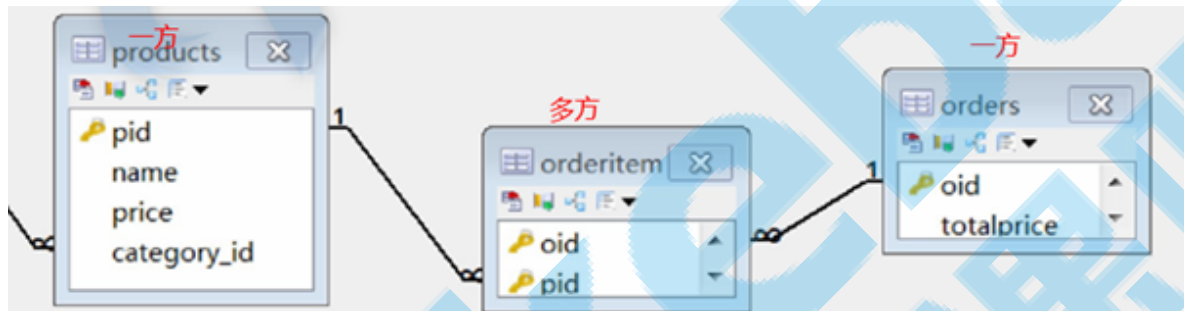
- 添加外键列
- 添加外键约束

案例：

1. 在商品表中添加一条记录，该记录的cid在分类表中不存在
2. 在分类表中，删除一条记录，这条记录在商品表中有外键关联

多对多关系

案例：同一个商品对应多个订单，一个订单对应多个商品



- 注意事项：

需要中间表去完成多对多关系的创建

多对多关系其实就是两个一对多关系的组合

- 多对多关系创建：

创建中间表，并在其中创建多对多关系中两张表的外键列

在中间表中添加外键约束

在中间表中添加联合主键约束

用户和角色

1个用户对多个角色

1个角色对多个用户

中间表用户角色表 `uid rid`

多表关联查询

我们已经学会了如何在一张表中读取数据，这是相对简单的，但是在真正的应用中经常需要从多个数据表中读取数据。

本章节我们将向大家介绍如何使用 MySQL 的 JOIN 在两个或多个表中查询数据。

你可以在 SELECT, UPDATE 和 DELETE 语句中使用 MySQL 的 JOIN 来联合多表查询。

`JOIN` 按照功能大致分为如下三类：

- `CROSS JOIN`（交叉连接）
- `INNER JOIN`（内连接或等值连接）。
- `OUTER JOIN`（外连接）

交叉连接

关键字：

`CROSS JOIN`

交叉连接也叫笛卡尔积连接。笛卡尔积是指在数学中，两个集合 X 和 Y 的笛卡尔积（Cartesian product），又称直积，表示为 $X \times Y$ ，第一个对象是 X 的成员而第二个对象是 Y 的所有可能有序对的其中一个成员。

交叉连接的表现：

行数相乘、列数相加

- 隐式交叉连接

```
SELECT * FROM A, B
```

```
mysql> select * from product,category;
```

pid	pname	price	pdate	cid	id	cname
1	泰国大榴莲	98	NULL	1	1	国外食品
1	泰国大榴莲	98	NULL	1	2	国内食品
1	泰国大榴莲	98	NULL	1	3	国内服装
2	泰国大枣	38	NULL	1	1	国外食品
2	泰国大枣	38	NULL	1	2	国内食品
2	泰国大枣	38	NULL	1	3	国内服装
3	新疆切糕	68	NULL	2	1	国外食品
3	新疆切糕	68	NULL	2	2	国内食品
3	新疆切糕	68	NULL	2	3	国内服装
4	十三香	10	NULL	2	1	国外食品
4	十三香	10	NULL	2	2	国内食品
4	十三香	10	NULL	2	3	国内服装
5	泰国大枣	20	NULL	2	1	国外食品
5	泰国大枣	20	NULL	2	2	国内食品
5	泰国大枣	20	NULL	2	3	国内服装
6	泰国大枣	98	NULL	20	1	国外食品
6	泰国大枣	98	NULL	20	2	国内食品
6	泰国大枣	98	NULL	20	3	国内服装
7	iPhone手机	800	NULL	30	1	国外食品
7	iPhone手机	800	NULL	30	2	国内食品
7	iPhone手机	800	NULL	30	3	国内服装

21 rows in set (0.00 sec)

- 显式交叉连接

```
SELECT * FROM A CROSS JOIN B
```

内连接

关键字：

```
INNER JOIN
```

内连接也叫**等值连接**，内联接使用比较运算符根据每个表共有的列的值匹配两个表中的行。

- 隐式内连接

```
SELECT * FROM A,B WHERE A.id = B.id
```

```
mysql> select * from product a , category b where a.cid=b.id;
```

pid	pname	price	pdate	cid	id	cname
1	泰国大榴莲	98	NULL	1	1	国外食品
2	泰国大枣	38	NULL	1	1	国外食品
3	新疆切糕	68	NULL	2	2	国内食品
4	十三香	10	NULL	2	2	国内食品
5	泰国大枣	20	NULL	2	2	国内食品

5 rows in set (0.00 sec)

- 显式内连接

```
SELECT * FROM A INNER JOIN B ON A.id = B.id
```

外连接

外联接可以是左向外联接、右向外联接或完整外部联接。也就是说外连接又分为：

左外连接、右外连接、全外连接

外连接需要有**主表或者保留表**的概念。

在 `FROM` 子句中指定外联接时，可以由下列几组关键字中的一组指定：

- 左外连接：

LEFT JOIN 或者 LEFT OUTER JOIN

```
SELECT * FROM A LEFT JOIN B ON A.id = B.id
```

mysql> select * from product a left outer join category b on a.cid=b.id; # 以左表数据为准

pid	pname	price	pdate	cid	id	cname
1	泰国大榴莲	98	NULL	1	1	国外食品
2	泰国大枣	38	NULL	1	1	国外食品
3	新疆切糕	68	NULL	2	2	国内食品
4	十三香	10	NULL	2	2	国内食品
5	泰国大枣	20	NULL	2	2	国内食品
6	泰国大枣	98	NULL	20	NULL	NULL
7	iPhone手机	800	NULL	30	NULL	NULL

7 rows in set (0.00 sec)

- 右外连接:

RIGHT JOIN 或者 RIGHT OUTER JOIN

```
SELECT * FROM A RIGHT JOIN B ON A.id = B.id
```

mysql> select * from product a right outer join category b on a.cid=b.id; #以右表数据为准

pid	pname	price	pdate	cid	id	cname
1	泰国大榴莲	98	NULL	1	1	国外食品
2	泰国大枣	38	NULL	1	1	国外食品
3	新疆切糕	68	NULL	2	2	国内食品
4	十三香	10	NULL	2	2	国内食品
5	泰国大枣	20	NULL	2	2	国内食品
NULL	NULL	NULL	NULL	NULL	3	国内服装

6 rows in set (0.00 sec)

- 全外连接 (MySQL不支持) :

FULL JOIN 或 FULL OUTER JOIN

```
SELECT * FROM A FULL JOIN B ON A.id = B.id
```

外连接总结：

- 通过业务需求，分析主从表
- 如果使用 `LEFT JOIN`，则主表在它左边
- 如果使用 `RIGHT JOIN`，则主表在它右边
- 查询结果以主表为主，从表记录匹配不到，则补 `null`

