

MySQL关系型数据库

(一) 数据库基本概念

作者 : Daniel.Wang





课程主要内容

- 1. 数据库相关概念
- 2. MySQL安装与配置
- 3. 库管理
- 4. 表管理
- 5. 结构化查询语言
- 6. 数据约束
- 7. 数据导入导出
- 8. 权限管理
- 9. 数据库事务
- 10. 存储引擎
- 11. 性能调优
- 12. E-R关系图
- 13. Python访问MySQL



课程总体目标

- 理解、掌握关系数据库基本理论、重要概念
- 熟练掌握MySQL安装、配置、管理
- 熟练掌握库、表的管理（新增、修改、删除）
- 熟练掌握数据增、删、改、查操作
- 掌握权限、数据备份/恢复等日常管理操作
- 掌握E-R设计工具
- 熟练使用PyMySQL库，实现Python对MySQL数据库的操作



课程特点

- 入门容易，提高难
- 知识点较多、较小，并且较为零散
- 无法看见数据库底层实现原理



为什么要学习数据库

- 数据库是开发人员必备知识
- 软件离开数据库就无法实现功能



(一) 数据库概述



1. 什么是数据库

1) 数据库(Database): 根据某种数据模型进行组织，并存放到计算机存储设备的数据集合。笼统来讲，就是存放数据的仓库

2) 数据库管理系统 (DBMS : DataBase Management System) :

› 定义：位于操作系统和用户之间的专门进行数据管理的软件系统

› 常见的DBMS : Oracle, MySQL, DB2, SQL Server, Informix

3) 数据库系统：一般性统称，包含DBMS、数据库软硬件设备、应用程序、DBA、用户



2. 数据库应用场合

1) 数据库是一种重要的基础软件 , 几乎应用于所有的软件系统

2) 示例 :

- 银行客户信息、账户信息、交易信息存储
- 电子商务网站商品、订单、客户信息存储
- 仓库中所有物品信息、数量、位置的存储
- 新闻系统新闻内容、图片、视频的存储
- 论文网站收录所有文献
-



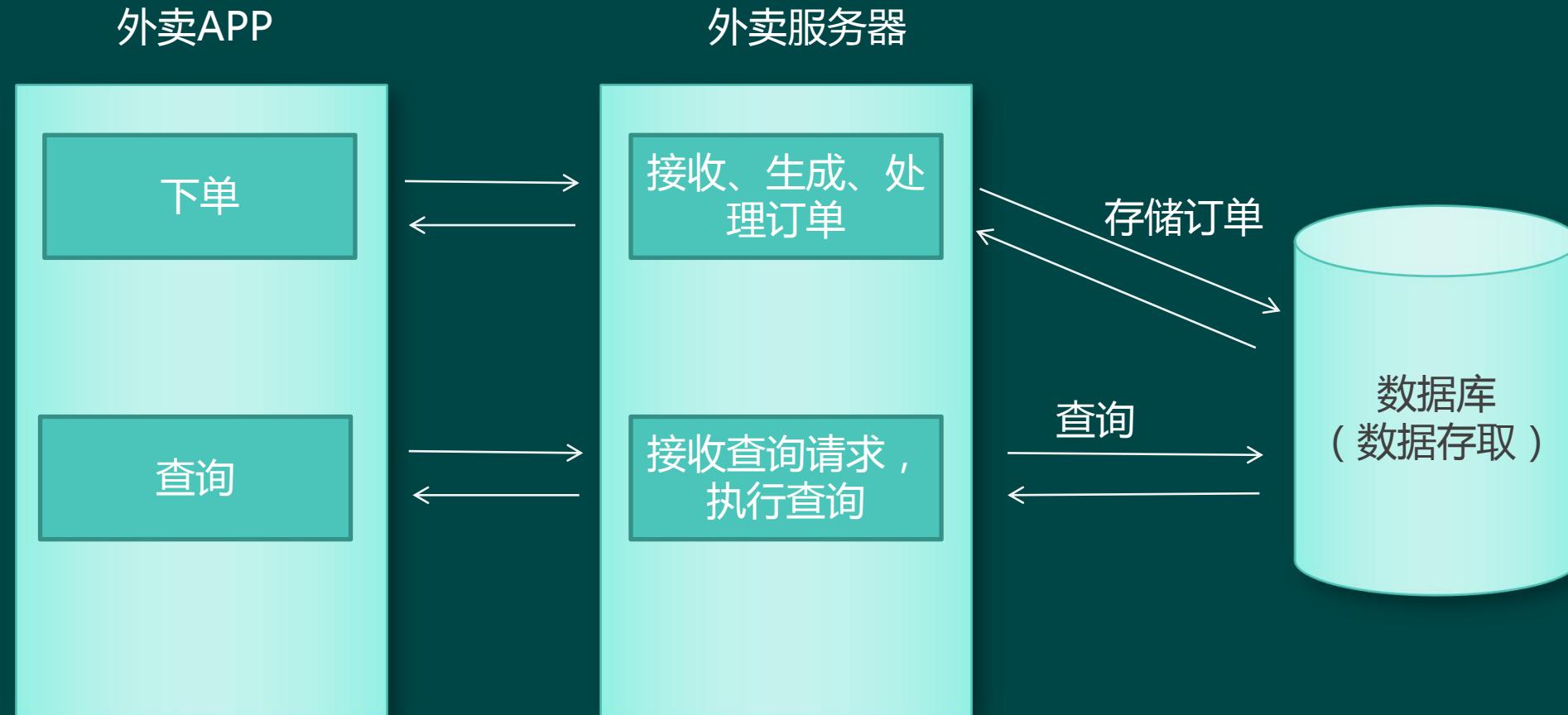
3. 业界主流的DBMS

厂商	数据库管理系统
Oracle	Oracle, MySQL
Microsoft	SQL server, Access
IBM	DB2
Sybase	Sybase
加州大学伯克利分校	PostgreSQL

* 前三个厂商占市场份额85%以上



4. 数据库在软件中的地位



思考：

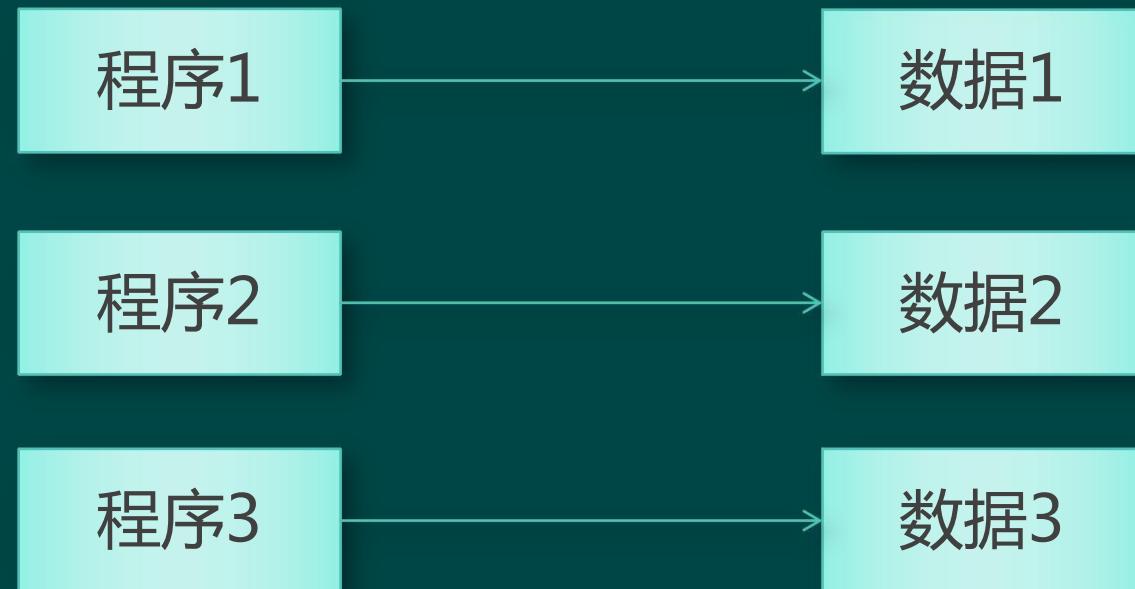


数据库最早出现于上世纪60年代，计算机是1946年诞生的，在数据库出现之前，数据是如何管理的，存放在哪里？



5. 数据管理的三个阶段

1) 人工管理阶段 : 计算机不通过数据管理 ; 程序和数据是不可分割的整体 ; 数据不能共享 ; 不单独保存数据 ;





2) 文件系统阶段 : 数据以文件形式持久保存在外部存储设备上

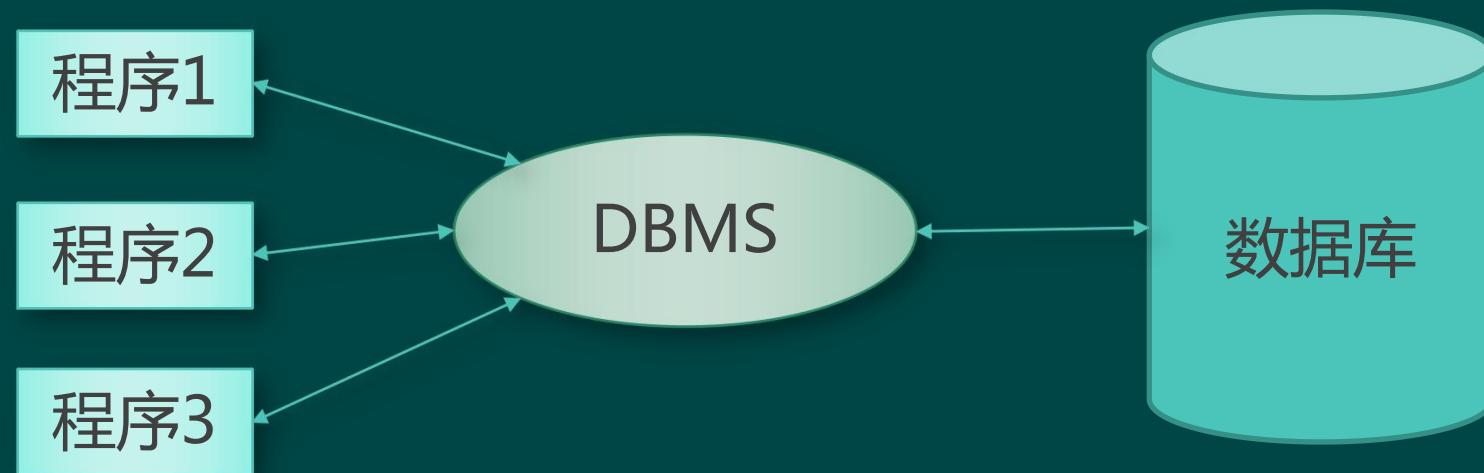
- 优点 : 程序和数据实现了分离 ; 数据的逻辑结构和物理结构有了区别 ; 文件的建立 , 数据增、删、改、查都要用程序来实现 ;
- 缺点 : 数据冗余 ; 不一致性 ; 数据联系弱 ;





3) 数据库管理阶段 : 单独使用一套软件系统来存储、管理数据

- 优点 :
 - ✓ 数据独立性、可共享、低冗余
 - ✓ 数据库系统提供了方便、友好的接口 (用户接口、程序接口)
 - ✓ 更强的安全性、可靠性
 - ✓ 丰富的工具 (如性能优化、备份/恢复、查询、权限管理)
- 缺点 : 需付出额外的软件、硬件、人力成本





6. 数据库概念模型

1) 层次模型

2) 网状模型

3) 关系模型

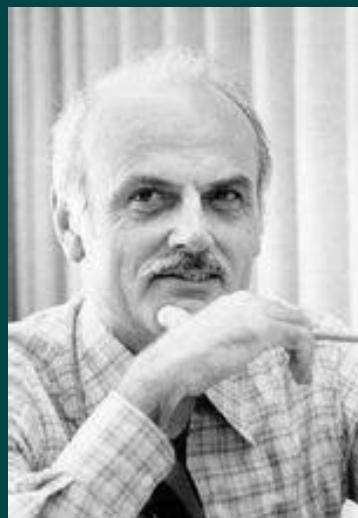
- 目前主流数据库模型
- 使用二维表（关系）表示数据、数据之间的联系

4) 非关系模型



关系模型产生

1970年，IBM研究院E.F.Codd在《大型共享数据库的关系模型》中提出了关系模型的概念，后来又发表多篇文章，奠定了关系数据库的基础。由于关系模型简单明了、具有坚实的数学理论基础，所以一经推出就受到了学术界和产业界的高度重视和广泛响应，并很快成为数据库市场的主流。



关系模型奠基人
E.F.Codd



Oracle创始人
拉里·埃里森



关系模型的特点

- 1) 建立在严格的数据理论基础上
- 2) 概念单一、简单、结构清晰，操作方便
- 3) 能很好保证数据的一致性、完整性



7. 关系模型主要概念

1) 关系 (Relationship)

- 由行、列组成的规范的二维表（每个属性不能重名，关系中每一行次序不重要）
- 行描述一个实体（事物）信息，列表示实体的属性



关系描述实体、实体间联系示例

- 订单信息表（描述了两笔订单信息）

订单编号	客户编号	下单时间	商品数量	订单总金额
0001	C0001	2018-01-01 10:02:54	1	100.00
0002	C0002	2018-01-02 12:02:54	3	340.00

- 客户信息表（描述了两个客户信息）

客户编号	客户证件类型	客户证件号码	客户姓名	客户电话
C0001	身份证件	513823197501011111	张三	13512345678
C0002	护照	E0451234	托马斯·李	13122334455



- 订单明细表

订单编号	下单时间	商品编号	商品数量
0001	2018-01-01 10:02:54	P0001	1
0002	2018-01-02 12:02:54	P0002	1
0002	2018-01-02 12:02:54	P0003	2

- 订单信息表：描述一组订单信息
- 客户信息表：描述一组客户信息
- 订单明细表：描述各订单详细信息
- 可以通过订单表中客户编号，找到客户信息（订单-客户发生了联系）；可以通过订单编号，找到订单详情（订单-订单明细发生了联系）



2) 关系数据库 : 使用关系 (二维表) 实体和实体间的联系的数据库管理系统

3) 实体 (Entry) : 现实中可以区分的事物称为实体

4) 元组 (Tuple) : 表中的一行 (也叫一条记录) , 表示一个实体

5) 属性 (Attribute) : 表中的一列 , 描述实体的一个数据值

6) 键 (Key) : 关系中唯一区分不同元组的属性或属性组合

7) 主键 (Primary Key) : 多个键中选取一个作为主键

› 作用 : 关系中 (二维表) 从逻辑上唯一确定一个实体的依据

› 满足条件 : 一个表最多只能有一个主键 ; 非空、不重复



(二) MySQL简介

1. MySQL基本情况

1) 最为著名、应用最广泛的开源数据库软件

- ›最早隶属于瑞典的MySQL AB公司
- ›2008年1月 MySQL AB被Sun收购
- ›2009年4月 Sun被Oracle收购



2) 崭新的开源分支 MariaDB

- ›为应付MySQL可能会闭源的风险而诞生
- ›由MySQL原作者Widenius主导开发
- ›与MySQL保持最大程度兼容





2. MySQL的特点

- 1) 开源 , 成本低
- 2) 体积小、速度快
- 3) 支持Linux/Unix、 windows等主流操作系统
- 4) 使用C和C++编写 , 可移植性强
- 5) 支持丰富的开发语言接口 , C、C++、Python、Java、PHP、.NET.....



3. MySQL应用场景

- 1) 几乎能用于所有的使用关系数据库软件系统中
- 2) 主要是中小企业、网站



4. MySQL主要版本

- 1) MySQL Community Server 社区版本，开源免费，但不提供官方技术支持
- 2) MySQL Enterprise Edition 企业版本，需付费，可以试用30天
- 3) MySQL Cluster 集群版，开源免费。可将几个MySQL Server封装成一个Server
- 4) MySQL Cluster CGE 高级集群版，需付费



(三) MySQL安装及配置



1. MySQL的安装

1) Windows系统安装

- 第一步：下载安装文件 mysql-installer-community-5.7.24.0.msi
- 第二步：点击安装
 - ✓ 选 server only或developer default
 - ✓ 设置端口：保持默认3306
 - ✓ 设置root用户名
 - ✓ 添加用户，并设置密码
- 第三步：确认
 - ✓ 在命令行模式下输入：netstat -an | findstr 3306
 - ✓ 查看服务端口是否处于监听状态



MySQL Windows系统目录结构

目录	目录内容
bin	客户端程序与mysql服务端
data	日志文件、数据库
docs	文档
include	包含(头)文件
lib	库
share	错误信息文件



2) Ubuntu系统安装

➤ 第一步 :

- ✓ `sudo apt-get install mysql-server`
- ✓ `sudo apt-get isntall mysql-client`
- ✓ `sudo apt-get install libmysqlclient-dev`

➤ 第二步 : 确认

- ✓ 在命令行模式下输入 : `netstat -an | grep 3306`
- ✓ 通过脚本查看 : `sudo /etc/init.d/mysql [参数]`

status: 查看服务状态

start: 启动服务

stop: 停止服务

restart: 重启服务



MySQL Ubuntu下目录结构

目录	目录内容
/usr/bin	命令及可执行文件
/var/lib/mysql/	数据存储目录
/etc/init.d/mysql	服务管理脚本（启动、停止服务，查看服务状态）
/etc/mysql	服务配置文件
/usr/share/mysql	错误消息、字符集、示例配置文件等等

* 服务器配置主要放在 /etc/mysql/mysql.conf.d 文件中



3) MySQL组成

- 客户端-服务器：由客户端、服务器量大部分组成
- 服务器（ mysqld ）：数据库的核心，处理数据保存、读写大部分操作
- 客户端（ mysql ）：用户进行登录、发出指令、显示结果；用户一般情况下不直接操作服务器，而是通过客户端登录、发出各种指令来操作服务器，从而实现各种功能





4) 使用客户端登录服务器

指令 : mysql -hlocalhost -P3306 -uroot -p123456

参数说明 :

- -hlocalhost : 连接服务器地址 , 如果不输则使用localhost
- -uroot : 使用用户root登录
- -p123456 : 用户密码123456
- -P3306 : 连接绑定端口 , 如果不输则默认使用3306
- 退出 : exit, quit



客户端更多参数说明：

参数	描述
-D , --database=name	打开指定数据库
--delimiter = name	指定分隔符
-h , --host=name	服务器名称
-p , --password[=name]	密码
-P , --port=#	端口号 (MySQL默认端口号为3306)
--prompt=name	设置提示符
-u , --user=name	用户名
-V , --version	输出版本信息并且退出



2. SQL语言简介

1) SQL语言 : **结构化查询语言** (Structured Query Language) , 它是一种操作数据库的工具语言 , **用于DBMS中数据增删改查、数据库管理。**

1986年10月 , 美国国家标准协会对SQL进行规范后 , 以此作为关系式数据库管理系统的标准语言 (ANSI X3. 135-1986) , 1987年得到国际标准组织的支持下成为国际标准。不过各种通行的数据库系统在其实践过程中都对SQL规范作了某些编改和扩充。所以 , 实际上不同数据库系统之间的SQL不能完全相互通用。



2) SQL语句使用规则

- 每条SQL语句必须以分号结束
- 不区分大小写
- 不支持Tab键自动补齐
- 使用\c可废弃当前写错的SQL指令



(四) MySQL库管理



1. 查看库

1) 查看库命令 : `show databases;` (列出服务器上所有库)

2) MySQL系统级库

- `information_schema`:数据库元数据的访问方式，比如字符集，权限相关，数据库实体对象信息，外检约束，分区，压缩表，表信息，索引信息，参数，优化，锁和事物等等
- `performance_schema`:收集数据库服务器性能参数
- `sys`:所有的数据源来自`performance_schema`，目标是把`performance_schema`的复杂度降低，让DBA能更好的阅读这个库里的内容
- `mysql`: 系统级表，例如存储用户、权限等信息



2. 创建库

1) 创建库命令 :

```
create database 库名称 [default charset=字符集]
```

2) 示例 : 创建名为eshop的库

```
create database eshop default charset=utf8
```

3) 库的命名规则

- 可以使用数字、字母、_，但不能使用纯数字
- 库名区分字母大小写
- 库名具有唯一性
- 不能使用特殊字符和MySQL关键语法



数据库的构成

数据库其实就是一个容器，它由表、视图、索引、触发器、存储过程、用户等对象组成，这些对象称为数据库对象。所以在使用这些对象之前，必须先行创建数据库



3. 进入/切换库

1) 进入库命令 :

use 库名称

2) 示例 : 进入eshop库

use eshop;



4. 查看库

1) 查看当前所在库 :

```
select database();
```

2) 查看当前库中的表 :

```
show tables;
```

3) 查看某个库的建库语句 :

```
show create database 库名称
```

如 : show create database eshop



5. 删除库

1) 删除库命令 :

```
drop database 库名;
```

2) 示例 : 删除eshop库

```
drop database bank;
```



(五) MySQL表管理



1. 创建表

1) 创建表必须先进入某个库

2) 建表语法 :

```
create table 表名称(  
    字段1 类型(长度) 约束,  
    字段2 类型(长度) 约束,  
    .....  
) [字符集];
```



3) 示例：创建订单表，包含订单编号、客户名称字段

```
create table orders (
    order_id varchar(32),
    cust_name varchar(32)
) default charset=utf8;
```



2. 查看表结构

1) 语法 : desc 表名称

2) 示例 : 查看orders表的结构

desc orders;

```
mysql> desc orders;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| order_id | varchar(32) | YES |   | NULL    |       |
| cust_name | varchar(32) | YES |   | NULL    |       |
+-----+-----+-----+-----+-----+
2 rows in set (0.10 sec)
```



3. 查看建表语句

1) 语法 : show create table 表名称

2) 示例 : 查看orders表的建表语句

```
show create table orders;
```

```
mysql> show create table orders;
+-----+
| Table | Create Table
+-----+
| orders | CREATE TABLE `orders` (
  `order_id` varchar(32) DEFAULT NULL,
  `cust_name` varchar(32) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-----+
1 row in set (0.00 sec)
```



4. 删除表

1) 语法 : drop table 表名称

2) 示例 : 删除orders表

```
drop table orders;
```



5. 示例：重新创建订单表

```
create table orders (
    order_id varchar(32),      -- 订单编号
    cust_id varchar(32),        -- 客户编号
    order_date datetime,        -- 下单日期时间，日期时间类型
    status int,                -- 订单状态，整数型
    products_num int,          -- 商品数量，整数型
    amt decimal(16,2)           -- 订单总金额，最大16位，小数部分2位
) default charset=utf8;
```



容易出错的地方：

- SQL语句中出现中文符号
- 括号必须匹配、嵌套正确，最好成对编写
- date不是data
- 注意其它拼写错误
- 最后一个字段后面没有分号，语句的分号不要忘记



(六) MySQL表记录管理



1. 插入数据

1) 向表中插入一笔数据 , 所有字段都插入值

```
insert into orders
```

```
values('201801010001', 'C0001', now(), 1, 1, 100.00);
```

说明 :

- ✓ 如果省略字段列表 , 那么必须为所有字段赋值
- ✓ 字段值的数目、顺序、数据类型必须与建的数目、顺序、数据类型完全匹配
- ✓ 空值用 NULL 表示
- ✓ now()表示取系统当前时间
- ✓ 字符类型必须用单引号引起来



2) 向表中插入指定字段数据

- ◆ 语法 : INSERT INTO 表名(字段名列表) VALUES(值列表)
- ◆ 示例 : 向订单表(orders)中插入一笔数据 , 只插入订单编号(order_id)、客户编号 (cust_id)

```
insert into orders(order_id, cust_id) values('201801010002', 'C0002');
```

- ◆ 说明 :
 - ✓ 如果指定了字段 , 那么值的数目、顺序、数据类型必须与指定字段的数目、顺序、数据类型完全匹配
 - ✓ 未插入值的字段 , 值为NULL

order_id	cust_id	order_date	status	products_num	amt
201801010002	C0002	NULL	NULL	NULL	NULL



3) 向表中插入多笔数据

- ◆ 语法 :

INSERT INTO 表名(字段名列表) VALUES(值列表1), (值列表2)...(值列表N)

- ◆ 示例 :

```
insert into orders values
```

```
('201801010003', 'C0003', now(), 1, 2, 200.00),  
('201801010004', 'C0004', now(), 1, 3, 480.00);
```



2. 查询数据

1) 语法格式

`select * from 表名称 [where 查询条件]`

`select 字段1, 字段2 from 表名称 [where 查询条件]`

2) 示例

- ✓ 查询所有数据、所有字段

`select * from orders;`

- ✓ 查询指定字段

`select order_id, order_date from orders;`

- ✓ 查询指定字段，给每个字段一个别名

`select order_id "订单编号", order_date "下单时间" from orders;`



✓ 带条件的查询

```
select * from orders where order_id = '201801010001';
```

✓ 带多个条件的查询

```
select * from orders
```

```
where order_id = '201801010001'
```

```
and status = 1;
```

两个条件同时满足

```
select * from orders
```

```
where order_id = '201801010001'
```

```
or order_id = '201801010002' ;
```

满足两个条件中的一个



(七) MySQL数据类型



1. 常见信息种类

- 1) 数值类型 : 身高、体重、成绩、金额.....
- 2) 字符类型 : 名称、地址、密码、电话、电子邮件.....
- 3) 日期时间类型 : 交易日期、注册日期、发生时间.....
- 4) 枚举类型 : 性别、爱好



2. 数值类型

类型	大小	范围(有符号)	范围(无符号)	用途
TINYINT	1字节	(-128, 127)	(0, 255)	小整数值
SMALLINT	2字节	(-32 768, 32 767)	(0, 65 535)	大整数值
MEDIUMINT	3字节	(-8 388 608, 8 388 607)	(0, 16 777 215)	大整数值
INT或 INTEGER	4字节	(-2 147 483 648, 2 147 483 647)	(0, 4 294 967 295)	大整数值
BIGINT	8字节	(-9 233 372 036 854 775 808, 9 223 372 036 854 775 807)	(0, 18 446 744 073 709 551 615)	极大整数值
FLOAT	4字节	(-3.402 823 466 E+38, 1.175 494 351 E-38), 0, (1.175 494 351 E-38, 3.402 823 466 351 E+38)	0, (1.175 494 351 E-38, 3.402 823 466 E+38)	单精度 浮点数值
DOUBLE	8字节	(1.797 693 134 862 315 7 E+308, 2.225 073 858 507 201 4 E-308), 0, (2.225 073 858 507 201 4 E-308, 1.797 693 134 862 315 7 E+308)	0, (2.225 073 858 507 201 4 E-308, 1.797 693 134 862 315 7 E+308)	双精度 浮点数值
DECIMAL	对DECIMAL(M,D) , 如果M>D, 为M+2 否则为D+2	依赖于M和D的值	依赖于M和D的值	小数值



数值类型示例

```
create table num_test(  
    type int(3) unsigned zerofill,-- 显示3位,无符号 , 左边0填充  
    rate decimal(10,2)  
);  
  
insert into num_test values(1, 0.88);      -- 正常值  
insert into num_test values(2, 123.456);    -- 浮点部分超长 , 四舍五入  
insert into num_test values(3,2);          -- 浮点数字段插入整数  
insert into num_test values(1000,3.444);    -- 整数部分超宽 , 全部显示
```



数值类型示例说明

- ✓ 当使用unsigned修饰类型时，字段的值只能是正数
- ✓ 定义字段时指定的长度仅仅为显示宽度，存储值得大小由数据类型决定
- ✓ 当使用ZEROFILL属性时，左边会以0补位
- ✓ 数值超过范围会报错
- ✓ 浮点数字段插入整数值时，自动用0填补小数部分
- ✓ 小数部分超过指定长度，自动四舍五入



3. 字符类型

1) 定长 : CHAR

最大存储255个字符，如果不足255个字符，右边以空格填充

如果不指定字符数，默认字符数为1

超过长度，无法存入

2) 变长 : VARCHAR

最大字符数为65535

按数据实际大小分配存储空间

字符数超出时则无法写入数据

3) 大文本类型 : TEXT

字符数大于65535存储时使用，最大能保存4GB的字符



CHAR和VARCHAR的特点

CHAR类型浪费存储空间，但是性能高

VARCHAR类型节省存储空间，但是性能低

推荐使用VARCHAR



4. 枚举类型

- 1) ENUM : 从给定的几个值中选择一个值
- 2) SET : 从给定的几个值中选择一个或多个值
- 3) 示例 :

```
create table enum_test(  
    name varchar(32),  
    sex enum('boy', 'girl'),  
    course set('music', 'dance', 'paint', 'football')  
);  
insert into enum_test values('Jerry', 'girl', 'music,dance');  
insert into enum_test values('Tom', 'boy', 'football');  
insert into enum_test values('Dekie', 'boy', 'football,bascketball');-- 超出枚举范围 , 报错
```



5. 日期时间类型

- 1) 日期 : DATE, 范围 '1000-01-01' ~ '9999-12-31'
- 2) 时间 : TIME
- 3) 日期时间 : DATETIME, 范围 '1000-01-01 00:00:00' ~ '9999-12-31 23:59:59'
- 4) 时间戳 : TIMESTAMP



日期时间函数

类 型	用 途
now()	获取调用此函数时的系统日期时间
sysdate()	执行时动态获得系统日期时间
sleep(N)	休眠N秒
curdate()	获取当前的系统日期
curtime()	获取当前的系统时刻
month()	获取指定时间中的月份
date()	获取指定时间中的日期
time()	获取指定时间中的时刻

日期时间函数示例



- ✓ 获取系统当前时间

```
select now(),sysdate();
```

- ✓ 获取系统当前日期，时间

```
select curdate(), curtime();
```

- ✓ 取得系统当前时间的年份、月份、日

```
select year(now()), month(now()), day(now());
```

- ✓ 将当前系统时间转换为日期、时间类型

```
select date(now()), time(now());
```



(八) 总结与回顾



基本概念

- 数据库：根据某种模型，对数据集中存放、管理的仓库
- 数据库管理系统：位于操作系统和用户之间的专门进行数据管理的软件系统
- 业界主流DBMS：Oracle, MySQL, SQL Server, DB2
- 数据管理三个阶段：人工管理阶段，文件管理阶段，数据库管理阶段
- 数据库概念模型：层次模型，网状模型，关系模型，非关系模型



关系模型重要概念

- 关系：规范的二维表，由行和列组成；每一列都不可再分，表中行顺序不重要
- 实体：现实中可以区分的事物
- 元组：二维表中的一行，每个元组记录一个实体信息
- 属性：二维表中的一列，描述实体的某个特征
- 键：关系中唯一区分不同元组的属性或属性组合
- 主键：从逻辑上唯一确定一个实体，多个键中选取一个作为主键，一个关系只能有一个主键；主键非空、唯一



MySQL的特点

- 广泛应用的关系型数据库
- 免费、开源
- 可移植性好
- 速度快，体积小
- 多应用与中小企业、互联网行业中



MySQL服务器的管理

- 查看状态 : /etc/init.d/mysql status
- 启动 : /etc/init.d/mysql start
- 停止 : /etc/init.d/mysql stop



MySQL库管理

- 查看库 : show databases;
- 进入库 : use 库名;
- 查看当前所在的库: select database();
- 创建库 : create 库名称 default charset=字符集;
- 删除库 : drop database 库名;
- 查看库中的表: show tables;

MySQL表管理



- 创建表

```
create table 表名称(
```

```
    字段1 类型(长度) 约束,
```

```
    字段2 类型(长度) 约束,
```

```
.....
```

```
) [字符集];
```

- 查看表结构: desc 表名称;

- 查看创建指定表的详细语句 : show create table 表名称;

- 删除表 : drop table 表名称;

MySQL数据操作



➤ 插入

```
insert into orders values('201801010001', 'C0001', now(), 1, 1, 100.00);
```

```
insert into orders(order_id, cust_id) values('201801010002', 'C0002');
```

➤ 查询

```
select * from orders;
```

```
select * from orders where order_id = '201801010001' ;
```

```
select * from orders where order_id = '201801010001' and cust_id = 'C0002';
```

```
select * from orders where order_id = '201801010002' or cust_id = '201801010002';
```



MySQL主要数据类型

- 数值类型：整型(TINYINT, SMALLINT, INT, BIGINT)
- 字符类型：定长char, 变长varchar
- 日期时间类型：日期、时间、时间戳
- 枚举类型：enum(多个值中选一个), set(多个值中选一个或多个)
- 日期时间函数：
 - now(), sysdate()
 - curdate(), curtime()
 - year(), month(), day()
 - date(), time()