**mysql**

# 数据及数据库

## 数据:对客观事物进行记录并且可以鉴别的符号。

### 模拟数据:连续的值。声音,图像等

### 数字数据:离散的值。符号,文字等

## 数据存储：纸张，磁盘，光盘，硬盘等。

## 数据存储的目的:检索。

## 数据库(DB:database):按照一定数据结构存储数据的仓库。

## 数据分类:

### 结构化数据:能够使用统一的结构表示的数据。如数字，字符等

### 半结构化数据:介于完全结构化和完全无结构化之间。如xml和html等

### 非结构化数据:长度可变，不能使用统一结构表示的数据。如图像，音频等

## 数据库分类

### 关系型数据库:使用二维表格存储结构化数据。

### 非关系型数据库(Nosql):存放半结构化和非结构化数据。

## DBMS(database management system):数据库管理系统。用于操纵和管理数据库的软件系统。

## 数据库管理系统的分类

### 关系型数据库管理系统(RDBMS:Relation database management system)

#### Oracle: Oracle公司的大型分布式数据库管理系统，产品免费，服务收费。

#### Mysql:Oracle公司,开源免费，速度快，支持GPL协议。

#### SQL Server: Microsoft公司，运行于windows平台

#### DB2：IBM公司

### 非关系型数据库管理系统:

#### Redis

#### MongoDB

#### Hbase apache hadoop

# mysql数据库

## 简介

瑞典Mysql AB公司开发的关系型数据库管理系统。

MySQL是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL所使用的 SQL 语言是用于访问[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93" \t "_blank)的最常用标准化语言。体积小，速度快，成本低，支持千万级数据，成为中小型网站开发的主流选择。

1996年发布mysql 1.0

2000年开源

2008年被sun以10亿美元收购

2009年被sun被oracle以74亿美元收购

## 下载安装

[https://dev.mysql.com/downloads/](https://dev.mysql.com/downloads/installer/)

安装版:

绿色版:

### 解压压缩包到英文路径

### 在cmd中进入bin目录

### 安装mysql服务: mysqld –install

### 启动服务: net start mysql

### 使用services.msc查看服务是否启动

### 停止服务:net stop mysql

### 移除服务:mysqld –remove

### 配置环境变量:path中配置mysql的路径

## 连接mysql服务

### mysql –uroot –p密码 默认无密码

## 常用命令

### show databases; 显示所有数据库

### use dbname; 指定使用某个数据库

### show tables; 显示该数据库下所有的表。

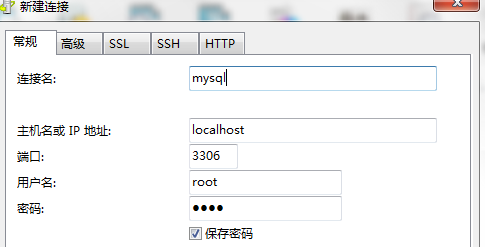
### \h：提示使用命令

### \q,exit:退出连接

### mysqladmin –uroot –p password 新密码:修改数据库密码

## 客户端工具

navcat,sqlyog



# sql简介

sql(structured query language):结构化查询语言。用于操作和管理数据的语言。

# mysql的数据类型

## 数值类型

### TINYINT 1

### SMALLINT 2

### MEDIUMINT 3

### INT 4

### BIGINT 8

### float(n,m) n代表长度，m代表小数点后位数

### double(n,m)

### decimal(n,m)

## 日期类型

### date:日期

### time:时间

### datetime:日期时间

### timestamp:时间戳

### year：年

## 字符串类型

### char:定长字符串

### varchar:可变长字符串

### binary：定长二进制

### varbinary：可变长二进制

### blob:二进制

### text:大对象，字符，文本

### enum:枚举

### set:集合

# sql的分类

## DDL(data definitation language):数据定义语言

数据库，表，视图等的创建和销毁

create，drop

## DML(data manipulation language):数据操纵语言

数据的增删改操作

insert:添加

delete:删除

update:修改

## DQL(data query language):数据查询语言

数据的查询

select

## DCL(data control language):数据控制语言

授权和取消授权

grant,revoke

# 数据库的操作

## 创建数据库

### 语法: create {database|schema} [if not exists] dbname;

|  |
| --- |
| 创建数据库  create database if not exists mydb; |

## 销毁数据库

### 语法:drop {database|schema} [if exists] dbname;

|  |
| --- |
| 销毁数据库  drop database if exists mydb;  []表示可省略 |

# 数据库表的操作

## 数据库中的数据存放在数据库表中，以二维数组的形式存在。

一行代表一条数据记录，称为记录。

一列代表同一域的数据，表示同一属性。称为字段。

## DDL操作

### 创建:create table [if not exists] tname;

|  |
| --- |
| 创建数据库表  create table `user`(  userid int,  username varchar(20),  `password` varchar(20),  birthday date  ); |

### 查看表结构:desc tname;

### 修改表结构: alter

#### 添加字段

alter table tname add [column] col\_name col\_definition [after|first]

|  |
| --- |
| 添加sex列(默认最后一列)  alter table `user` add sex varchar(10);  添加到指定列后  alter table `user` add sex varchar(10) after userid;  添加到首列  alter table `user` add sex varchar(10) first; |

#### 修改字段

alter table tname change col\_name new\_col\_name col\_definition;

|  |
| --- |
| 修改数据类型  alter table user CHANGE sex sex VARCHAR(10);  修改字段名称  alter table user CHANGE sex usex VARCHAR(10);  修改字段位置  alter table tuser change birthday birthday date after userid; |

#### 删除字段

alter table `user` drop col\_name;

|  |
| --- |
| 删除sex列  alter table user drop sex; |

#### 重命名

rename table tname to new\_tname;

|  |
| --- |
| 修改user表为tuser  rename table user to tuser; |

#### 清空表

|  |
| --- |
| 清空重置表所有数据，速度比delete快。自增将重置。  truncate table tuser; |

## 完整性约束

### 完整性即数据的准确性。

### 实体完整性:保证记录是唯一的，不重复的。

主键约束:primary key

|  |
| --- |
| 主键字段唯一且不能为空   * 创建表时声明主键   create table user(  userid int primary key,  username varchar(20)  );   * 创建表时声明主键   create table user(  userid int,  username varchar(20),  primary key(userid)  );   * 修改表结构添加约束   alter table user add CONSTRAINT PK\_USERID primary key(userid); |

唯一约束:unique

|  |
| --- |
| 唯一约束字段值不能重复   * 创建表时声明唯一约束   create table user(  userid int primary key,  username varchar(20),  card varchar(18) unique,  );   * 创建表时添加唯一约束   create table user(  userid int primary key,  username varchar(20),  card varchar(18),  unique(card)  );   * 修改表结构添加唯一约束   alter table user add CONSTRAINT UN\_CAED unique(card); |

主键自增:auto\_increment

|  |
| --- |
| create table user(  userid int primary key auto\_increment,  username varchar(20),  card varchar(18)  ); |

### 域完整性：保证列的数据正确性。

类型约束

非空约束：not null

|  |
| --- |
| create table user(  userid int primary key auto\_increment,  username varchar(20) not null,  card varchar(18) not null  ); |

默认值约束

|  |
| --- |
| create table user(  userid int primary key auto\_increment,  username varchar(20) not null,  sex varchar(10) default '男',  card varchar(18) not null  ); |

### 引用完整性：保证引用关系(外键)的正确性，参照完整性

|  |
| --- |
| alter table emp add CONSTRAINT PK\_DEPTNO foreign key(deptno) REFERENCES dept(deptno) |

### 自定义完整性(check约束mysql不支持)

|  |
| --- |
| 添加约束保证部门编号只能在1到120之间  alter table dept add CONSTRAINT CK\_AGE check(deptno BETWEEN 1 and 120) |

### 删除约束

|  |
| --- |
| alter table emp drop primary key;  alter table emp drop foreign key; |

## 运算符

### 算术运算符 + - \* / div整除

|  |
| --- |
| select 1+2;  select 1-2;  select 3\*4;  select 3/4;/\*0.75\*/  select 3/0;/\*null\*/  select 3 div 4; |

### 比较运算符 >,<,>=,<=,!=,<>

Is true,is not true,is null,is not null,between and,not between and,,in, not in

|  |
| --- |
| select 1<>1;/\*0\*/  select 1!=1;/\*0\*/  select 1=1;/\*1\*/ |

### 逻辑运算符 and or !

|  |
| --- |
| select 1=1 and 1=2;/\*0\*/  select !1<>1;/\*1\*/  select 1=1 or 1=2;/\*1\*/ |

### 位运算符

|  |
| --- |
| select 1 | 1;  select 1^2;  select 1&0; |

## DML操作

### 添加数据 insert

#### 插入数据

|  |
| --- |
| values中的数据值与表的字段位置一致,字符串插入需要使用引号  insert into user values(12,'zs','123'); |

#### 向特定字段位置插入数据

|  |
| --- |
| 前提是某些字段允许空值操作  insert into user(userid,username) values(3,'ww'); |

#### 批量插入数据

|  |
| --- |
| insert into user values(1,'zs','123'),(2,'ls','123') |

#### 复制已有的表

|  |
| --- |
| insert into user(id,name) select userid,username from tuser |

### 修改数据 update

update tname set col\_name = col\_value

|  |
| --- |
| 使用update没有使用条件，将修改所有记录的值  update tuser set password = '234' where userid = 2; |

### 删除数据 delete

|  |
| --- |
| 使用delete语句，没有where条件将删除所有的数据  //删除所有数据是一行一行删除，因此比truncate速度慢，而且不会重置自增值。  delete from tuser where userid = 1; |

## DQL操作 select

select 的语法

select \* from tbl\_name

FROM 子句

where 子句

group by

HAVING

order by

|  |
| --- |
| * 查询所有数据   select \* from user   * 查询指定列的数据   select username,password from user   * 条件查询(单一):查询年龄>18的用户信息   select \* from user where age > 18   * 条件查询(组合),多个条件使用and或or连接   查询年龄>18的所有女生  select \* from user where age >18 and sex = '女'   * 别名(字段)   select userid [as] 编号,username [as] 名称 from user;   * 别名(表)   select u.userid,u.username from user u;   * 去重(distinct)   select distinct username from user;   * 模糊查询 like   %代表匹配0到多个字符  \_代表匹配一个字符  查询以z打头的所有用户  select \* from user where username like 'z%';  查询以s结尾的所有用户  select \* from user where username like '%s';  查询包含s的所有用户  select \* from user where username like '%s%';  查询第二个字为s的用户  select \* from user where username like '\_s%'   * 非空 is null   select \* from user where sex is [not] null;   * 范围查询 between … and …:查询年龄在12和20之间的用户   select \* from user where age BETWEEN 12 and 20;  等价于  select \* from user where age >=12 and age <=20;   * 集合查询：查询用户id为1和2的用户   select \* from user where userid in (1,2);  等价于  select \* from user where age = 1 or age =2;   * 排序:按照年龄升序排序   select \* from user order by age [asc]  按照年龄升序排序,如果年龄一样，按照编号降序排序  select \* from user order by age,userid desc;   * 限制结果数量:用于分页查询 limit m,n:m代表开始索引(从0开始),n代表长度   select \* from user limit 0,2; |

# 函数

## 单行函数

### 数学函数

|  |
| --- |
| * abs():绝对值   select abs(-1); //1   * ceiling()/ceil():向上取整,最接近并且大于等于该值的整数值   select ceil(12.5); //13  select ceil(-12.5); //12   * floor():向下取整,最接近并且小于等于该值的整数值   select floor(12.5); //12  select floor(-12.5); //13   * mode(m,n):m对n取模   select mode(5,3); //2   * PI():取PI的值 * pow(m,n):取m的n次方   select POWER(3,2); //9   * rand():取随机数   select rand(); //[0,1)之间获取随机数   * round(m,n):四舍五入   select round(3.5); //4  select ROUND(13.56,2); //13.56 从小数点后n为开始四舍五入   * truncate(m,n):截取m小数点后n位   select TRUNCATE(3.56,1); //3.5 |

### 字符函数

|  |
| --- |
| * ascii(str):获取str的ascii码值   select ASCII('a'); //97   * lower(字段|表达式):将字符串转换为小写   select LOWER('ABC'); //abc   * upper(字段|表达式):将字符串转换为大写   select UPPER('abc'); //ABC   * concat(str1,str2…):将字符串连接   select CONCAT('aa',2,'cc'); //aa2cc   * length(字段|表达式):获取字符串长度   select LENGTH('hello mysql'); //11   * substr(str,pos,len):截取字符串,pos开始位置，从1开始；len表示长度   select SUBSTR('my name is gardy\_lee',4,4); //name   * replace(str,old,new):在str中搜索old，使用new代替   select REPLACE('my name is gardy\_lee','my','your');//your name is grady\_lee   * lpad(str,len,s):str长度不够len，使用s左侧填充   select LPAD('hello',10,'--'); //-----hello   * rpad(str,len,s):str长度不够len,使用s右侧填充   select RPAD('hello',10,'-'); //hello-----   * trim():去重左右两侧的空格   select trim(' hello'); //hello |

### 日期函数

|  |
| --- |
| * NOW()/SYSDATE()/CURRENT\_TIMESTAMP():获取当前日期时间 * CURRENT\_DATE()/CURDATE():获取当前系统日期 * CURRENT\_TIME()/CURTIME():获取当前系统时间 * DATE\_ADD(date,INTERVAL expr unit):日期转换   select ADDDATE('1998-08-08',INTERVAL 2 YEAR); //2000-08-08   * DAY(date):获取天数   select DAY(‘1998-09-10’); //10   * MONTH(date):获取月份   select MONTH('1989-09-10'); //9   * YEAR(date):获取年份   select YEAR('1989-09-10'); //1989   * week(date):返回一年中的周数   select WEEK('2001-02-05'); //5   * weekday(date):返回一周中的第几天(0-6)   select WEEKDAY('2001-02-06'); //1 周二 |

## 聚合函数

|  |
| --- |
| * avg():平均值 * sum():获取总数 * max():获取最大值 * min():获取最小值 * count():统计数目 |

## 分组函数

|  |
| --- |
| * GROUP BY子句   + 用于将信息划分为更小的组   + 每一组行返回针对该组的单个结果 * HAVING子句   + 用于指定 GROUP BY 子句检索行的条件   查询平均工资大于2000的部门编号  select deptno,avg(sal) avg from emp group by deptno having avg>=2000 |

## 加密函数

|  |
| --- |
| * select md5('root'); * select SHA('root'); * select PASSWORD('root'); |

# 高级查询

## 多表查询(关联查询，连接查询)

### 笛卡尔积

emp表15条记录,dept表4条记录。

连接查询的笛卡尔积为60条记录。

### 内连接:不区分主从表，与连接顺序无关。两张表均满足条件则出现结果集中。

|  |
| --- |
| * where子句   select \* from emp,dept  where emp.deptno = dept.deptno   * inner join…on…   select \* from emp  inner join dept  on emp.deptno = dept.deptno   * inner join…using…   select \* from emp  INNER JOIN dept  using(deptno) |

### 自然连接:寻找两表中字段名称相等的字段进行连接，会自动去重重复列。

|  |
| --- |
| select \* from emp NATURAL join dept; |

### 外连接:有主从表之分，与连接顺序有关。以驱动表为依据，匹配表依次进行查询；匹配表中找不到数据，则以null填充。

#### 左外连接:left [outer] join…on…

|  |
| --- |
| select \* from emp  LEFT JOIN dept  on emp.deptno = dept.deptno |

#### 右外连接:right [outer] join…on…

|  |
| --- |
| select \* from dept  LEFT JOIN emp  on emp.deptno = dept.deptno |

## 子查询:即嵌套查询，将一个查询结果作为另一个查询条件或组成部分的查询。

|  |
| --- |
| * 单行子查询:查询工资大于7788号员工的所有员工信息   select \* from emp where sal >(select sal from emp where empno =7788);   * 多行子查询:   返回多值可以使用any或all来修饰。  =any相当于in,<any小于最大值，>any大于最小值；  <>all相当于not in,>all大于最大值，<all小于最小值。   * 查询超过所在部门平均工资的员工信息   select \* from emp e1 where sal >  (select avg(sal) from emp e2 where e1.deptno = e2.deptno);   * 查询薪水大于2000的部门名称   select dname from dept d where deptno in  (select deptno from emp e where sal > 2000);  或  select dname from dept d where EXISTS  (select \* from emp e where sal > 2000 and d.deptno=e.deptno);   * in和exists的区别   in先执行子查询，在执行主查询;exists先执行主查询;  exists子查询不返回具体结果，返回true值出现在结果集，否则不出现。 |

## 集合查询

### UNION：并集，所有的内容都查询，重复的显示一次

### UNION ALL：并集，所有的内容都显示，包括重复的

|  |
| --- |
| select \* from emp where deptno = 20  UNION ALL  select \* from emp where sal <=2000; |

# 事务

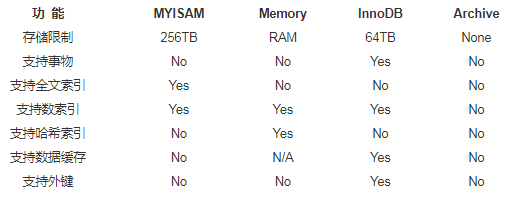
## 存储引擎(show ENGINES;)

数据库存储引擎是数据库底层软件组织，数据库管理系统（DBMS）使用数据引擎进行创建、查询、更新和删除数据。不同的存储引擎提供不同的存储机制、索引技巧、锁定水平等功能，使用不同的存储引擎，还可以获得特定的功能。Mysql的核心就是存储引擎。

InnoDB是事务型数据库的首选，执行安全性数据库，行锁定和外键。mysql5.5之后默认使用。

MyISAM插入速度和查询效率较高，但不支持事务。

MEMORY将表中的数据存储在内存中，速度较快。



## 什么是事务

事务用于保证数据的一致性，由一组DML操作组成，该组SQL语句要么同时成功，要么同时失败。例如转账。

## 事务的ACID特性

|  |
| --- |
| A(Atomicity)原子性:事务是原子工作单元，要么同时执行，要么同时不执行。  C(Consistency)一致性:符合约束规则；执行之前的整体状态和执行后数据一致。  I(Isolation)隔离性:并发事务之间相互不影响。  D(Durability)持久性:事务完成之后，对数据库的影响是永久的。  //开启事务  set Autocommit=0;  start TRANSACTION;  update user set money = money -100 where userid = 1;  SAVEPOINT a;//设置保存点  update user set money = money + 100 where userid = 2;  rollback to a;//回滚到保存点状态  commit;//提交事务，提交事务所有的保存点全部失效  rollback;//回滚事务 |

# 存储程序

## 存储程序指的一组存储和执行在数据库服务器端的程序。存储程序总是在服务器的进程或者线程的内存中执行的。

## 存储程序分为：

存储过程：有输入输出参数，可以执行一组sql指令。

存储函数：有一个返回值，可以对sql进行有效扩展。

触发器：指事件响应，比如执行insert语句后执行另外一个动作。

## 存储过程

|  |
| --- |
| * 创建存储过程(不带参数)   create procedure sel\_user()  BEGIN  select username from user;  end;  //调用存储过程  call sel\_user();   * 创建带参数的存储过程   create PROCEDURE sel\_user\_id(uid int)  BEGIN  select \* from user where userid = uid;  end;  //调用存储过程  call sel\_user\_id(1);   * 参数类型:in输入参数，out输出参数，inout输入输出参数   create PROCEDURE sel\_user\_id( INOUT count int)  BEGIN  select sal into count from emp where empno = count;  end;  //调用存储过程  set @count = 7788;  call sel\_user\_id(@count);  select @count;   * 变量   create PROCEDURE sel\_user()  BEGIN  -- 声明变量  DECLARE v\_name varchar(20);  -- 设置变量值  set v\_name = 'zs';  select v\_name;  end;  //调用存储过程  call sel\_user();   * 控制语句   CREATE PROCEDURE test(IN v\_count int,OUT v\_name VARCHAR(20))  BEGIN  IF v\_count > 10 THEN  set v\_name = 'zs';  ELSEIF v\_count > 5 THEN  set v\_name = 'ls';  ELSE  set v\_name = 'ww';  END IF;  END;  //调用存储过程  call test(12,@name);  select @name;   * while..do..end while/loop…end loop/repeat … util..end repeat   -- 使用while来计算1....10的和  create PROCEDURE my\_sum(n int,out m\_sum int)  BEGIN  DECLARE i int;  set i=0;  set m\_sum=0;  -- while 和do中间 加条件  while i<=n DO  set m\_sum=m\_sum+i;  set i=i+1;  end while;  end;  call my\_sum(10,@m\_sum);  select @m\_sum;  create PROCEDURE my\_sum1(n int,out m\_sum1 int)  BEGIN  DECLARE i int;  set i=0;  set m\_sum1=0;  -- loop需加标记 方便leave时指定  lip:loop    set m\_sum1=m\_sum1+i;  set i=i+1;  if i>n THEN leave lip;  end if;  end loop lip;  end;  call my\_sum1(10,@m\_sum1);  select @m\_sum1;  drop PROCEDURE my\_sum2;  create PROCEDURE my\_sum2(n int,out m\_sum2 int)  BEGIN  DECLARE i int;  set i=0;  set m\_sum2=0;  -- loop需加标记 方便leave时指定  REPEAT  set m\_sum2=m\_sum2+i;  set i=i+1;  UNTIL i>n  end REPEAT;  end;  call my\_sum2(10,@m\_sum2);  select @m\_sum2; |

## 存储函数(语法与存储过程一致，只有输入参数，有返回值，作为语句一部分使用)

|  |
| --- |
| CREATE FUNCTION test(eno int)  RETURNS VARCHAR(20)  DETERMINISTIC  BEGIN  DECLARE v\_name varchar(20);  select ename into v\_name from emp where empno = eno;  return v\_name;  end;  //调用函数  select test(7788); |

## 触发器

|  |
| --- |
| CREATE  [DEFINER = { user | CURRENT\_USER }]  TRIGGER trigger\_name  trigger\_time trigger\_event  ON tbl\_name FOR EACH ROW  [trigger\_order]  trigger\_body  trigger\_time: { BEFORE | AFTER }  trigger\_event: { INSERT | UPDATE | DELETE }  trigger\_order: { FOLLOWS | PRECEDES } other\_trigger\_name  create TRIGGER add\_t1\_t2  after INSERT  on t1 FOR EACH ROW  BEGIN  insert t2(name) values('tt');  end; |

# 视图和索引

## 视图是一张虚拟表，并不在数据库中以存储数据值集的形式存在；在引用过程中依据基表动态生成。

|  |
| --- |
| create or REPLACE view emp\_dept  as  select empno,ename,e.deptno,dname from emp\_copy e,dept\_copy d  where e.deptno = d.deptno and e.deptno = 10; |

## with check option

如果创建的视图中带有where等条件，则with check option 子句可以保证让你只能在视图的条件之内对视图进行DML。

## 删除视图

drop view emp\_dept;

## 视图优点

### 安全：有的数据是需要保密的，那么直接将表给出来进行操作将会造成泄密，那么可以通过创建视图，把相应视图的权限给出来即可保证数据的安全。

### 高效：复杂的连接查询，每次执行时效率比较低，可以考虑新建视图，每次从视图中获取，将会提高效率。

### 定制数据

## 什么是索引

索引是供服务器快速在表中查询一行数据的数据库结构。

适合在数据量较大的表上添加索引。

适合在该基数列上添加索引。

索引列不能经常变化，索引维护需要花费时间。

|  |
| --- |
| //创建索引  create index ind\_name on user(userid);  //删除索引  drop index ind\_name on user |

# 数据库的设计和优化

## 数据库设计

### 数据库设计是基础，数据库优化是建立在设计基础之上的。好的数据库一定拥有好的设计。

### 数据库设计的目标是为用户和各种应用系统提供一个信息基础设施和高效的运行环境。

### 数据库的三大范式

#### 所有的域都应该是原子性的，即数据库表的每一列都是不可分割的原子数据项，而不能是集合，数组，记录等非原子数据项。

#### 第二范式在第一范式的基础之上更进一层。

#### 第二范式需要确保数据库表中的每一列都和主键相关，而不能只与主键的某一部分相关（主要针对联合主键而言）。

#### 也就是说在一个数据库表中，一个表中只能保存一种数据，不可以把多种数据保存在同一张数据库表中。

#### 第三范式(确保每列都和主键列直接相关,而不是间接相关)

### 数据库设计步骤

#### 需求分析阶段

#### 准确了解与分析用户需求（包括数据与处理）

#### 是整个设计过程的基础，是最困难、最耗费时间的一步

#### 概念结构设计阶段

#### 是整个数据库设计的关键

#### 设计数据库的E-R模型图，确认需求信息的正确和完整

#### 逻辑设计阶段

#### 将E-R图转换为逻辑模型(逻辑模型为关系模型则体现为多张表)

#### 应用数据库设计的三大范式进行审核

#### 设计外模式，建立视图

#### 物理设计阶段

#### 确定存取方法（例如索引的设定）

#### 确定存储结构（文件存放的位置等），并编写代码实现前端应用

#### 数据库实施阶段

#### 将数据载入，并对数据库进行调试

#### 运行和维护阶段

#### 使用和维护数据库

### E-R图也称实体-联系图(Entity Relationship Diagram)

### 提供了表示实体类型、属性和联系的方法，用来描述现实世界的概念模型。

### 在ER图中有如下四个成分：

### 矩形框：表示实体，在框中记入实体名。

### 菱形框：表示联系，在框中记入联系名。

### 椭圆形框：表示实体或联系的属性，将属性名记入框中。对于主属性名，则在其名称下划一下划线。

### 连线：实体与属性之间；实体与联系之间；联系与属性之间用直线相连，并在直线上标注联系的类型。（对于一对一联系，要在两个实体连线方向各写1； 对于一对多联系，要在一的一方写1，多的一方写N；对于多对多关系，则要在两个实体连线方向各写N,M。)

## 数据库优化

### SELECT子句中避免使用(\*)ORACLE在解析的过程中, 会将’\*’ 依次转换成所有的列名, 这个工作是通过查询数据字典完成的, 这意味着将耗费更多的时间

### 索引失效的情况:

* + ① Not Null/Null 如果某列建立索引,当进行Select \* from emp where depto is not null/is null。 则会是索引失效。
  + ② 索引列上不要使用函数,
  + SELECT Col FROM tbl WHERE substr(name ,1 ,3 ) = ‘ABC’（不）
  + SELECT Col FROM tbl WHERE name LIKE ‘%ABC%’ （不）
  + SELECT Col FROM tbl WHERE name LIKE ‘ABC%’ （使用）。
  + ③ 索引列上不能进行计算
  + SELECT Col FROM tbl WHERE col / 10 > 10 则会使索引失效
  + 应该改成SELECT Col FROM tbl WHERE col > 10 \* 10
  + ④ 索引列上不要使用NOT （ != 、 <> ）
  + 如:SELECT Col FROM tbl WHERE col ! = 10 应该
  + 改成：SELECT Col FROM tbl WHERE col > 10 OR col < 10 。

### 用UNION替换OR(适用于索引列)

* + union:是将两个查询的结果集进行追加在一起，它不会引起列的变化。 由于是追加操作，需要两个结果集的列数应该是相关的，并且相应列的数据类型也应该相当的。
  + union 返回两个结果集，同时将两个结果集重复的项进行消除。 如果不进行消除，用UNOIN ALL.
  + 通常情况下, 用UNION替换WHERE子句中的OR将会起到较好的效果. 对索引列使用OR将造成全表扫描.
  + 注意, 以上规则只针对多个索引列有效.
  + 如果有column没有被索引, 查询效率可能会因为你没有选择OR而降低.

### 用EXISTS替代IN、用NOT EXISTS替代NOT IN

* + 在许多基于基础表的查询中, 为了满足一个条件, 往往需要对另一个表进行联接. 在这种情况下, 使用EXISTS(或NOT EXISTS)通常将提高查询的效率.
  + 在子查询中, NOT IN子句将执行一个内部的排序和合并. 无论在哪种情况下, NOT IN都是最低效的(因为它对子查询中的表执行了一个全表遍历).
  + 为了避免使用NOT IN, 我们可以把它改写成外连接(Outer Joins)或NOT EXISTS.
  + 高效: SELECT \* FROM EMP (基础表) WHERE EXISTS (SELECT ‘X’ FROM DEPT WHERE DEPT.DEPTNO = EMP.DEPTNO AND LOC = ‘MELB’)
  + 低效: SELECT \* FROM EMP (基础表) WHERE DEPTNO IN(SELECT DEPTNO FROM DEPT WHERE LOC = ‘MELB’)

# 备份和还原

Mysqldump -uroot -proot dbname > d:/t.msql;

mysql -u root -p < C:\backup.sql