mysql-review

知识点

1. 数据库概念

> - 数据(信息)：
>
>   数据(信息)是数据库中存储的基本对象,是关于现实世界事物的存在方式或运动状态反应的描述

- 数据库:

  是一个长期存储在计算机内的、有组织的、可共享的、统一管理的数据集合

- 数据库系统:

  数据库系统是指在计算机系统中引入数据库后的系统

- - 组成:

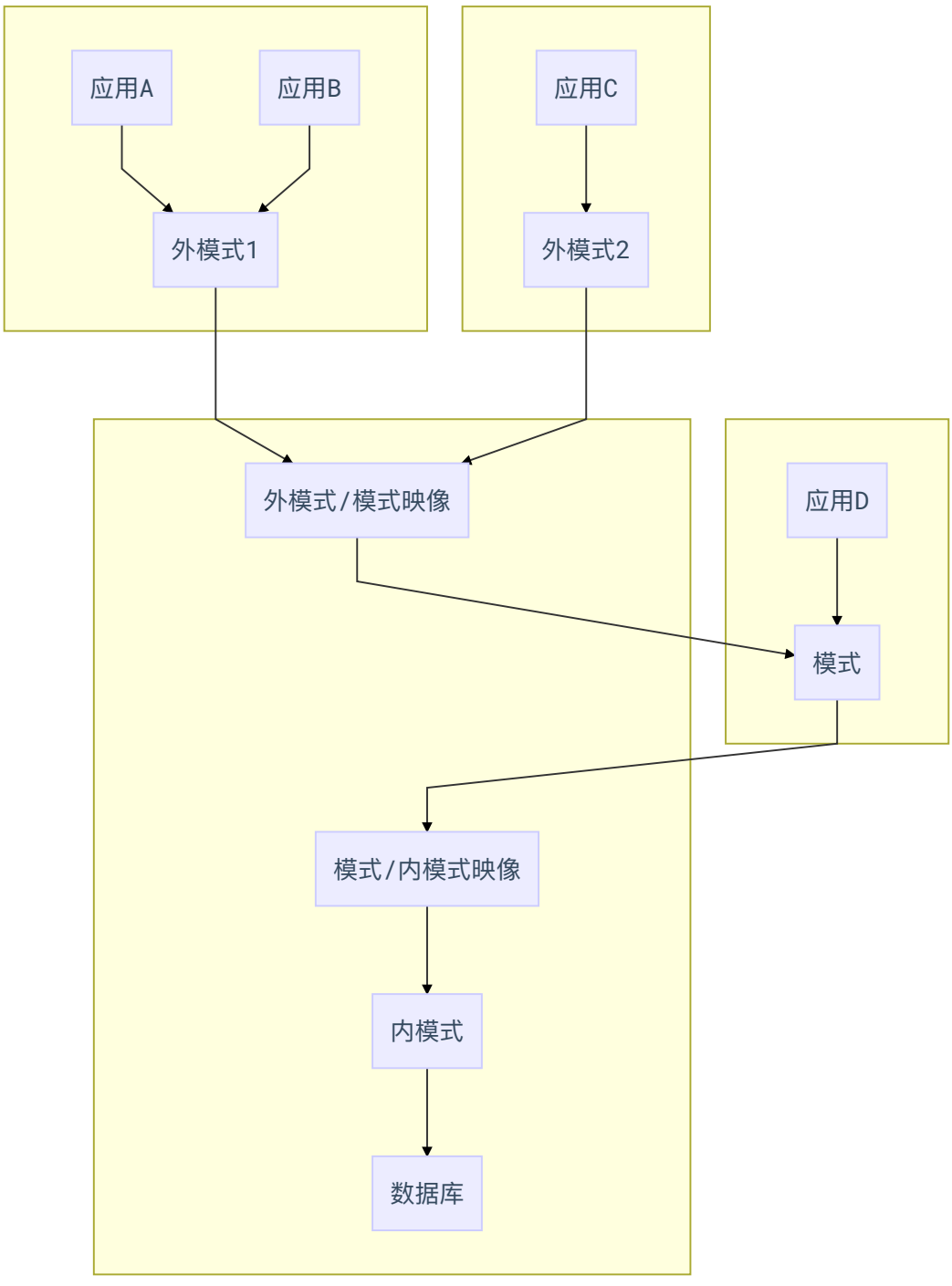
  数据库、软件系统和用户

2. 数据管理的三个阶段及特点

人工管理数据阶段	文件管理数据阶段	数据库管理阶段
(a) 数据一般不保存	(a) 数据可长期保存	(a) 数据结构化
(b) 没有专门的软件对数据进行统一管理	(b) 由文件系统对数据进行管理	(b) 数据共享性高、冗余小
(c) 数据无法共享	(c) 文件可多样化组织	(c) 数据独立性较高
(d) 数据不具有独立性	(d) `程序与数据间有一定的独立性	(d) 有统一的数据控制功能

文件管理与人工管理阶段比较未解决的根本性问题
1. 数据冗余度较大，数据一致性往往难以保证
2. 数据间的联系较弱
3. 数据独立性差
4. 缺乏对数据的统一控制

3. 数据库系统结构图


应用A
应用B
应用C
外模式1
外模式2
外模式/模式映像
应用D
模式
模式/内模式映像
内模式
数据库

结构与优点	
三级模式	概念模式、外模式、内模式
二级映像	外模式/模式映像、模式/内模式映像
优点	(a) 保证了数据的独立性
	(b) 有利于实现数据共享
	(c) 减轻了程序设计者的编程负担
	(d) 提高了数据安全性

4. 三个世界,两个抽象步骤

- 客观世界
- 信息世界
- 计算机世界

> ①将现实世界中客观对象抽象为概念模型。
>
> ②将概念模型抽象为某个DBMS所控制的数据模型。

5. ER图

6. 三个数据模型

- 层次模型:

  用树形结构表示实体及其之间的联系(可以直接方便地表示一对一和一对多联系)

- 网状模型:

  用网状结构表示实体及其之间联系的模型(可以直接表示多对多联系，但其中的节点间关系更加复杂)

- 关系模型:

> 用关系描述实体间的联系的数据模型(关系、元组、属性、域、分量、主码、关系模式)

7. 完整性约束三个条件

- 实体完整性规则
  > 指关系（所谓的关系就是表）的主码不能取空值
- 参照完整性规则
  > 指参照关系中每个元素的外码要么为空(NULL)，要么等于被参照关系中的某个元素的主码
- 用户自定义完整性规则
  > 指对关系中每个属性的取值作一个限制（或称为约束）的具体定义

8. 范式(Normal Form)

- 非范式：

  没有除去数据重复的表格
- 第一范式：

  如果关系模式R中每个属性值都是一个不可分解的数据项，则称该关系模式满足第一范式，简称1NF，记为R∈1NF
  > 将表格分割为单纯的二元表格，即一栏中只有一个项目，每一列都是不可分割的基本数据项
- 第二范式：

  如果一个关系模式R∈1NF，且它的所有非主属性都完全函数依赖于R的码，则R∈2NF
  > 按照通过可识别数据的键来确定其他列值的原则分割表格。这样，通过主键确定其他列的数值。 函数依赖 ： 通过某一列的值确定其他列的数值的原则
- 第三范式：

如果一个关系模式R∈2NF，且所有的非主属性都不传递函数依赖于码，则R∈3NF

> 按照只能由主键确定其他列值的原则分割的表格。 在关系数据库的函数依赖中,通过某一列的值间接确定其他列的值,称之为传递依赖。第三范式是去除传递依赖而分割表格得到的

9. ACID原理

- `A 原子(Atomiclty)` 事务原子性

  > 指每个事务都必须被看作是一个不可分割的单元

- `C 一致(Consistency)` 插入一张表数据，会影响其它(索引/其它表)等一致

  > 一致性属性保证了数据库从不返回一个未处理完的事务

- `I 隔离性(Isolation)` 事务独立，封闭；隔离性强度

  > 指每个事务在它自己的空间发生，和其他发生在系统中的事务隔离，而且事务的结果只有在它完全被执行时才能看到

- `D 持久性(Durability)` 数据永存

  > 指即使系统崩溃，一个提交的事务仍然存在

10. CAP原理

- `Consistency （一致性）`,数据一致更新，所有数据变动都是同步的。
- `Availability （可用性）`,好的响应性能。
- `Partition tolerance （分区容错性）` 可靠性。

> - CA 系统是要求高可用用并且实时一致性。单点数据库是符合这种架构的，例如超市收银系统，图书管理系统。
> - AP 满足可用性，分区容忍性的系统，通常可能对一致性要求低一些。例如博客系统。
> - CP 系统是要求满足一致性，分区容忍性，通常性能不是特别高。例如火车售票系

> 统。

MySQL

1. 数据库

```
/* 登录（DOS环境）*/
mysql -u root -p123456

/* 创建 */
create database school;

/* 查看 */
show databases;

/* 备份（DOS环境）*/
mysqldump -u root -p --databases school > d:\1108.sql

/* 删除 */
drop database school;

/* 还原 （DOS环境）*/
mysql -u root -p<d:\1108.sql
```

2. 表

```
/* 创建student表 */
create table student(
    sno char(10) primary key,
    sname varchar(6) not null,
    sgender enum('m','f') default 'f',
    saddress varchar(10),
    sqq char(10) unique
);

/* 创建course表 */
create table course(
    cno char(6) primary key,
    cname char(3) not null,
    ccredit tinyint default 3
);
```

```
16
17  /* 创建sc表 */
18  create table sc(
19      scno int primary key auto_increment,
20      sno char(10),
21      cno char(6),
22      grade float(4,2),
23      constraint fk_1 foreign key (sno) references   student(sno),
24      constraint fk_2 foreign key (cno) references
25      course(cno)
26  );
27
28  /* 查看student表结构 */
29  desc student;
30
31  /* 在student表里增加semail字段 */
32  alter table student add semail varchar(20);
33
34  /* 备份school数据库(DOS环境) */
35  mysqldump -u root -p --databases school > d:\1109.sql
36
37  /* 删除student表 */
38  alter table sc drop foreign key fk_1;
39  drop table student;
40
41  /* 还原数据库 */
42  mysql -u root -p < d:\1109.sql
```

3. 数据

```
1   /* 向student表插入数据 */
2   insert into student(sno,sname,sgender,saddress,sqq) values
3       ('2018001','tom','m','chn','12345'),
4       ('2018002','jim','m','usa',''),
5       ('2018003','mary','f','chn','54321')
6   );
7
8   /* 修改cname的数据长度为6 */
9   alter table course change cname cname char(6) not null;
10
11  /* 向course表插入数据 */
12  insert into course(cno,cname,ccredit) values
13      ('c001','math',3),
14      ('c002','db',4),
15      ('c003','os',2)
16  );
17
18  /* 向sc表插入数据 */
```

```
insert into sc(sno,cno,grade) values
    ('2018001','c001',88),
    ('2018001','c002',89),
    ('2018001','c003',90),
    ('2018002','c002',66),
    ('2018003','c001',70)
);

/* 修改tom的数学成绩为70.6分 */
update sc set grade=70.6
where sno in (select sno from student where sname='tom')
and cno = (select cno from course where cname='math');

/* 增加jim的数学成绩为60.8分 */
insert into sc(sno,cno,grade) values('2018002','c001',60.8);

/* 删除mary的数学成绩 */
delete from sc where sno='2018003' and cno='c001';

/* 备份数据库 */
mysqldump -u root -p --databases school > /home/1112.sql

/* 删除tom的学生信息 */
delete from sc where sno='2018001';
delete from student where sno='2018001';
```

4.查询

单表查询

```
/* 查询每个雇员的所有数据 */
select * from employees;

/*查询每个雇员的姓名、电话和地址*/
select name,address,phonenumber from employees;

/*查询EmployeeID为000001的雇员的地址和电话*/
select address,phonenumber from employees where employeeid="000001";

/*查询Employees表中女雇员的地址和电话,
使用AS子句将结果中各列的标题分别指定为地址、电话*/
select address as "地址",phonenumber as "电话" from employees where sex = "0";

/*查询Employees表中员工的姓名和性别,
要求gender值为1时显示为‘男’, 为0时显示为‘女’*/
select name as "姓名",
 case
  when sex="1" then "男"
```

```
    when sex="0" then "女"
   end as gender
  from employees;

  /*计算每个员工的实际收入*/
  select employeeid,income-outcome as "实际收入" from salary;

  /*获得员工总数*/
  select count(*) from employees;

  /*找出所有王姓员工的部门号*/
  select departmentid from employees where name like "王%";

  /*找出所有收入在2000~3000元之间的员工号码*/
  select employeeid from salary where income between 2000 and 3000;
```

多表查询

```
/* 查找在财务部工作的雇员的情况 */
select * from employees where departmentid = (select departmentid from
departments where departmentname="财务部");

/* 查找研发部年龄不低于市场部所有雇员年龄的雇员的姓名 */
select name from employees where departmentid in (select departmentid from
departments where departmentname="研发部") and birthday <= all(select birthday
from employees where departmentid in(select departmentid from departments where
departmentname="市场部"));

/* 查找比财务部所有的雇员收入都高的雇员的姓名 */
select name from employees where employeeid in (select employeeid from salary
where income > all(select income from salary where employeeid in (select
employeeid from employees where departmentid = (select departmentid from
departments where departmentname="财务部"))));
```

复杂查询

```
### 连接查询

/* 查询每个雇员的情况及其薪水的情况 */
select employees.*,salary.* from employees,salary where employees,salary where
employees.employeeid=salary.employeeid;

/* 使用内连接的方法查询名字为“王林”的员工所在的部门 */
select departmentname from departments join employees on
departments.departmentid=employees.departmentid where employees.name="王林";

/* 查找财务部收入在2000元以上的雇员姓名及其薪水详情 */
```

```
select name,income,outcome from employees,salary,departments where employees.employeeid=salary.employeeid and employees.departmentid=departments.departmentid and departmentname="财务部" and income>=2000;

### GROUP BY、ORDER BY和LIMIT子句的使用

/* 查找Employees中男性和女性的人数 */
select sex,count(sex) from employees group by sex;

/* 查找员工数超过2人的部门名称和员工数量 */
select departmentname,count(*) as "人数" from employees,departments where employees.departmentid=departments.departmentid group by employees.departmentid having count(*)>2;

/* 将Employees表中的员工号码由大到小排列 */
select employeeid from employees order by employeeid desc;

/* 返回Employees表中的前5位员工的信息 */
select * from employees limit 5;
```

5. 视图

```
### 创建视图

/* 创建YGGL数据库上的视图DS_VIEW，视图包含Departments表的全部列 */
create view ds_view as select * from departments;

/* 创建YGGL数据库上的视图Employees_view，视图包含员工号码、姓名和实际收入 */
create view employees_view(employeeid,name,realincome) as select employees.employeeid,name,income-outcome from employees,salary where employees.employeeid=salary.employeeid;

### 查询视图

/* 从视图DS_VIEW中查询出部门号为3的部门名称 */
select departmentname from ds_view where departmentid = "3";

/* 从视图Employees_view查询出姓名为"王林"的员工的实际收入 */
select realincome from employees_view where name="王林";

### 更新视图

/* 向视图DS_VIEW中插入一行数据：6，财务部，财务管理 */
insert into ds_view values("6","财务部","财务管理");

```

```
/* 修改视图DS_VIEW，将部门号为5的部门名称修改为“生产车间” */
update ds_view set departmentname="生产车间" where departmentid="5";

/* 修改视图Employees_view中号码为000001的雇员的姓名为“王浩” */
update employees_view set name="王浩" where employeeid="000001";

### 删除视图
/* 删除视图DS_VIEW中部门号为“1”的数据 */
delete from ds_view where departmentid="1";

/* 删除视图DS_VIEW */
drop view ds_view;
```

不能更新视图的条件

不能更新视图的条件
1 视图中包含sum()、count()、MAX()和MIN()等函数
2 视图中包含UNION、UNION ALL、DISTINCT、GROUP BY和HAVING等关键字
3 常量视图
4 视图中的SELECT中包含子查询
5 由不可更新的视图导出的视图
6 视图对应的表上存在没有默认值的列，而且该列没有包含在视图里

6. 索引

```
### 使用Create Index语句创建索引

/* 对YGGL数据库的Employees表中的DepartmentID列建立索引 */
create index depart_ind on employees(departmentid);

/*在Employees表的Name列和Address列上建立复合索引 */
create index ad_ind on employees(name,address);

/* 对Departments表上的DepartmentName列建立唯一性索引 */
create unique index dep_ind on departments(departmentname);

### 使用Alter Table语句向表中添加索引
```

```
/* 向Employees表中的出生日期列添加一个唯一性索引，姓名列和性别列上添加一个复合索引 */
alter table employees add unique index date_ind(birthday),add index
na_ind(name,sex);

/* 假设Departments表中没有主键，使用Alter Table语句将DepartmentID列设为主键 */
alter table employees add primary key(departmentid);

### 在创建表时创建索引

/* 创建与Departments表相同结构的表Departments1，将DepartmentName设为主
键,DepartmentID建立一个索引 */
create table departments1
(
    departmentid char(3),
    departmentname char(20),
    note text,
    primary key(departmentname),
    index did_ind(departmentid)
);

### 删除索引

/* 使用Drop Index语句删除表Employees上的索引 */
drop index depart_ind on employees;

/* 使用Alter Table语句删除Departments上的主键和索引 */
alter table departments drop primary key,drop index dep_ind;
```

索引设计的原则

索引设计的原则
选择唯一性索引
重复值越少，可以添加索引
为经常需要排序、分组和联合操作的字段建立索引
为常作为查询条件的字段建立索引
限制索引的数目
尽量使用数据量少的索引
尽量使用前缀来索引
删除不在使用或者很少使用的索引

7. 权限：创建用户、授权、收权

```
/* 使用root用户创建exam1用户，初始密码设置为123456 */
create user "exam1"@"localhost" identified by "123456";

/* 授权exam1用户对所有数据库拥有Select、Create、Drop和Grant权限 */
grant select,create,drop on *.* to "exam1"@"localhost" with grant option;

/* 查看exam1的权限 */
show grants for "exam1"@"localhost";
select * from mysql.user where user="exam1" and host="localhost";

/* 创建exam2，该用户没有初始密码 */
create user "exam2"@"localhost";

/* 用exam2登录，将其密码设置为787878 */
set password=password("787878");

/* 用exam1登录，为exam2授权为YGGL数据库的Employees表查看Name字段的权限 */
grant select(name) on yggl.employees to "exam2"@"localhost";

/* 用root登录，收回exam1和exam2的所有权限 */
revoke all privileges,grant option from
"exam1"@"localhost","exam2"@"localhost";
```

8. 触发器

```
/* 创建触发器，在Employees表中删除员工信息的同时将Salary表中该员工的信息删除，以确保数据完整性 */

create trigger delete_em after delete
    on employees for each row
    delete from salary
        where employeeid=old.employeeid;

/* 假设Department2表和Department表的结构和内容都相同，在Departments上创建一个触发器，如果添加一个新的部门，该部门也会添加到Departments2表中 */

delimiter $$
create trigger departments_ins after insert
    on departments for each row
    begin
        insert into departments2
values(new.departmentid,new.departmentname,new.note);
    end$$
delimiter ;
```

```
17
18  /* 当修改表Employees时，若将Employees表中员工的工作时间增加1年，则将收入增加500，增加2年则
    增加1000，依次增加。若工作时间减少则无变化 */
19
20  delimiter $$
21  create trigger add_salary after update
22      on employees for each row
23      begin
24          declare years integer;
25          set years=new.workyear-old.workyear;
26          if years>0 then
27              update salary set income=income+500*years
28                  where employeeid=new.employeeid;
29          end if;
30      end$$
31  delimiter ;
32
```

9. 存储过程、存储函数

存储过程

```
1   /* 创建存储过程，使用Employees表中的员工人数来初始化一个局部变量，并调用这个存储过程 */
2
3   delimiter $$
4   create procedure test(out number1 int)
5   begin
6       declare number2 int;
7       set number2=(select count(*) from employees);
8       set number1=number2;
9   end$$
10  delimiter;
11  call test(@number);
12  select @number;
13
14  /* 创建存储过程，比较两个员工的实际收入，若前者比后者高就输出0，否则输出1 */
15
16  delimiter $$
17  create procedure compa(in id1 char(6),in ide2 char(6),out bj int)
18  begin
19      declare sr1,sr2 float(8);
20      select income-outcome into sr1 from salary where employeeid=id1;
21      select income-outcome into sr2 from salary where employeeid=id2;
22      if id1>id2 then set bj=0;
23      else set bj=1;
24      end if;
25  end $$
26  delimiter;
```

```
call compa('00001','108991',@bj);
select @bj;

/* 创建存储过程，使用游标确定一个员工的实际收入是否排在前三名。结果为TRUE表示是，结果为FALSE表示否 */

delimter $$
create procedure top_three(in em_id char(6),out ok boolean)
begin
    declare x_em_id char(6);
    declare act_in,seq int;
    declare found boolean;
    declare salary_dis cursor for
        select employeeid,income-outcome from salary order by 2 desc;
    declare continue handle for not found set found=false;
    set seq=0;
    set found=true;
    set ok=false;
    open salary_dis;
    fetch salary_dis into x_em_id,act_in;
    while found and seq<3 and ok=false do
        set seq=seq+1;
        if x_em_id=em_id then set ok=true;
        end if;
        fetch salary_dis into x_em_id,act_in;
    end while;
    close salary_dis;
end $$
delimiter;
```

存储函数

```
/* 创建一个存储函数，返回员工的总人数 */

create function em_num()
    returns int
    return (select count(*) from employees);
select em_num();

/* 创建一个存储函数，删除在Salary表中有但在Employees表中不存在的员工号。若在Employees表中存在返回FALSE，若不存在则删除该员工号并返回TRUE */

delimiter $$
create function delete_em(em_id char(6))
    returns boolean
begin
    declare em_name char(10);
    select name into em_name from employees where employeeid=em_id;
    if em_name is null then
        delete from salary where employeeid=em_id;
```

```
18            return true;
19        else
20            return false;
21        end if;
22    end $$
23    delimiter ;
24    select delete_em('000001');
```