# mysql

## 1 目录

1. 数据库的认识
2. 单表操作
3. 多表操作入门
4. 多表操作加强
5. 数据库深入
6. 事务
7. 数据结构入门
8. 数据库优化及索引深入

## 2 数据库的认知

### 2-1 数据库的作用

**数据非常值钱。**

腾讯挣钱最轻松

腾讯最值钱的是： 数据，把全中国的人，关系网存下来了。

项目中，数据都保存在数据库中。

数据库，最基本，也是最重要的作用：保证数据的安全性

（删库跑路，腾讯最值钱的东西）

扩展：

计算机，各个硬件中，对性能影响最大的瓶颈是哪个硬件？

硬盘，最慢

数据，最终保存在硬盘是上面。

问题是：最值钱的数据，放在硬盘上的，但是，硬盘很慢。

**导致数据库很贵，可以快，又安全。**

在保证安全的基础上，还能提供优异的性能（速度），数据库就非常值钱

(oracle非常贵：https://www.cnblogs.com/ouyida3/p/4295381.html)

所以，oralce很贵。

### 2-2 加深理解

（短片）

琅琊阁的核心竞争力

信息的收集 信息的传递 信息的分析（整理） 信息的存储

各地的线人 信鸽 阁主，办公人员 溶洞（大架子）

映射到现在的互联网领域：

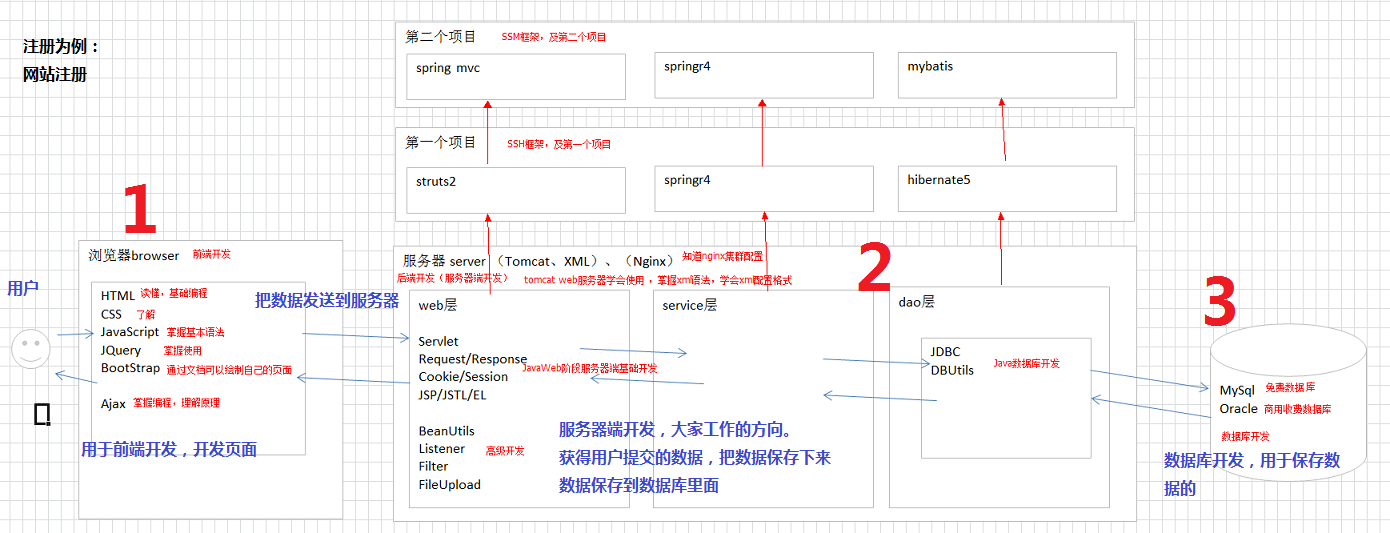
信息的收集 信息的传递 信息的分析（整理） 信息的存储

爬虫（用户自生产数据） 网络 大数据（数据挖掘，数据清洗，商业智能，数据仓库） 数据库

商业智能， 数据挖掘，决策系统，数据仓库。 都是建立在数据库保存数据的使用的基础上。

### 2-3 数据库的结构

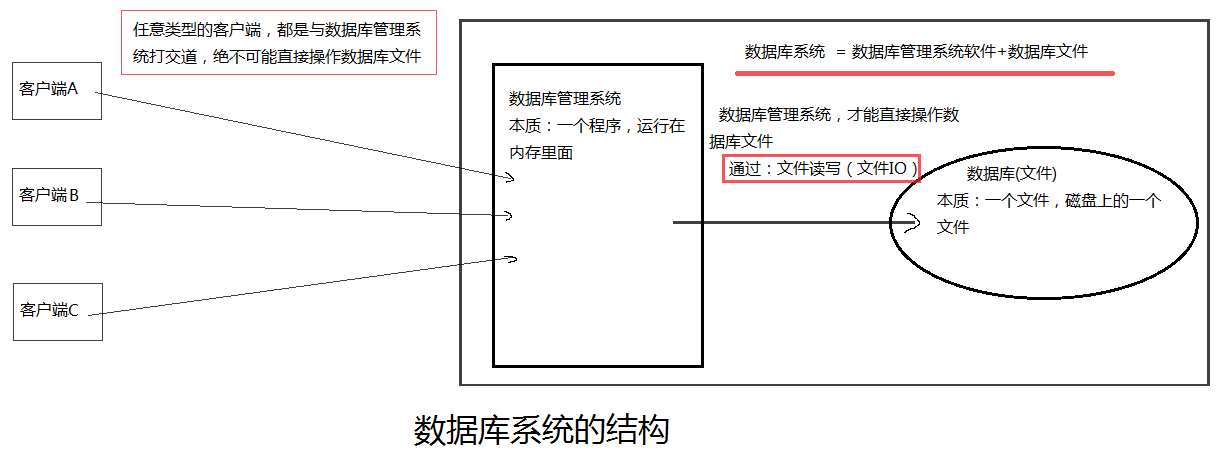
2-3-1 在项目中的位置



2-3-2 数据库的构成

\* 数据库系统 = 数据库管理系统（软件） + 数据库（文件）

\* 即： 我们口中的“数据库”，准确的描述是：数据库系统。



**第一个认知：数据库系统 = 数据库管理系统（软件） + 数据库（文件）**

因为：为了安全。

我们使用者，只是与数据库管理系统打交道，不会直接操作数据库文件。

### 2-4 相关概念

2-4-1 关系型数据库

1. 发展史

数据库分为层次模型、网状模型和关系模型三种类型。

关系模型是在1970年由IBM的研究员E.F.Codd博士首先提出的。在之后的几十年中，成为数据库结构的主流模型。

1. 关系型数据库，指采用了“关系模型”来组织数据的数据库，非常贴切实际。

类比：表格的行和列

2-4-2 sql语句

一门语言，专门用于操作关系型数据库的。

mysql，oracle, SQLServer，虽然是3个公司的数据库，但是都是关系型数据库，都支持sql语句，所以其操作基本相同。

2-4-3 NoSql数据库

不是指远古时代的层次模型，网状模型数据库。

而是进入互联网年代后，关系型数据库，无法应对新的需求（三高需求），而出现了一些不支持sql语言的数据库，例如：redis, MongoDB等

巨大的key-value集合。

|  |
| --- |
| 对数据库高并发读写的需求  对海量数据的高效率存储和访问的需求  对数据库的高可扩展性和高可用性的需求 |

重点关注：redis

## 3 单表操作

### 3-1 mysql和oracle内部存储的层次

mysql数据库内部分为3个层次：  
 数据库， 表， 表中记录

oracle数据库分为3个层次：  
 账号(用户)，表，表中记录

因为oracle有账号的概念，新增了一个功能： 权限管理

### 3-2 单表的12个基本操作

对3个层次：增删改查

其中：oracle，表归属于“账号”需要新建账号，并且要授权。

重点是：对表中记录的操作

扩展：实际开发

实际中，表已经建好，我们只需要了解业务相关的表即可

对记录的增删改查，只需要关注表**中记录的查询**

在实际的应用中，查询操作，远远大于增删改

所以，才理解，索引是神器。

索引的特性：提高查询的效率，降低增加删除修改的效率。

### 3-3 select查询语句

#### 3-3-1 7兄弟

|  |
| --- |
| 1. 语法   select 字段名(需要查询的属性，其中:\*，通配符，表示所有字段)  from 表名（需要查询的表）  where 条件表达式1(根据条件筛选记录)  group by 字段名1（按照那个字段分组）  having 条件表达式2（分组后，再次筛选）  order by 属性名2（查询后的结果，按照那个属性排序，默认：升序asc,如想降序是desc）  2. 第1个顺序是：语法顺序  select...from...where...group by...having...order by...      3. 第2个顺序是：执行顺序  a. from  b. where  c. group by  d. having  e. select  f. distinct  g. order by  4. 区别  \* 语法顺序：是格式规范，无条件遵守  \* 执行顺序：是编写查询语句的前提，必须认真理解。  \* 语法顺序和执行顺序，存在区别。 |

#### 3-3-2 where, group by, having 3个语句的理解。

要点：记住3个语句的执行顺序

a. where,group by, having三者的执行顺序

where： 第1个执行，即在group by分组之前执行

group by:第2个执行，按照指定的字段（属性）分组

having: 第3个执行，即在group by分组之后执行

b.where和having的区别

\* where：分组之前筛选，因没有分组，故where语句不能使用统计函数。

\* having:分组之后筛选，因已经分组，故having语句可以使用统计函数。

|  |
| --- |
| 案例：mysql   * 准备数据   create table star(pid int primary key auto\_increment, name char(10), sex char(2), age int,  income int, type char(4));    insert into star values(null, "袁隆平","男",88,30,"科学派");  insert into star values(null, "屠呦呦","女",88,32,"科学派");    insert into star values(null, "刘德华","男",56,12000,"实力派");  insert into star values(null, "成龙","男",64,16000,"实力派");  insert into star values(null, "刘涛","女",40,14000,"实力派");  insert into star values(null, "李宇春","女",34,8000,"实力派");    insert into star values(null, "杨幂","女",32,12400,"偶像派");  insert into star values(null, "杨颖","女",29,14000,"偶像派");  insert into star values(null, "吴亦凡","男",28,13600,"偶像派");    insert into star values(null, "文章","男",34,2000,"脱轨派");  insert into star values(null, "李小璐","女",37,1000,"脱轨派");  insert into star values(null, "贾乃亮","男",34,6000,"环保派");  insert into star values(null, "马伊琍","男",42,3000,"环保派");  insert into star values(null, "房祖名","男",36,3000,"坑爹派");  insert into star values(null, "张默","男",36,1000,"坑爹派");    insert into star values(null,"快递张","男",22,8,"平凡派");  insert into star values(null,"IT杨","男",30,24,"平凡派");  insert into star values(null,"服务员金","女",40,5,"平凡派");  \* 查询某类类别，该类明星的年收入超过2000的人数超过2人。  select type from star where income >2000 group by type having count(\*)>2;    \* 查询某类类别，该类明星的年平均收入超过2000，并且人数超过2人。  select type from star group by type having avg(income)>2000 and count(\*)>2;    \* 查询某类类别，该类明星的年收入超过2000的男性人数超过2人。  select type from star where income >2000 and sex ='男' group by type having count(\*)>2; |

#### 3-3-3 order by结合limit的妙用

|  |
| --- |
| 案例：查询男性中，工资排名前三的明星信息 |

select \*from star where sex='男' order by income desc limit 0,3;

## 4 多表操作入门

### 4-1 为什么要拆表

|  |
| --- |
| \* 实战模拟项目开发：学生管理系统  \* 用excel演示表的设计  1. 项目需求：保存学生的学号，姓名，年龄，成绩  2. 新增需求2：保存学生对应老师的姓名    3. 新增需求3：保存老师的职称，年龄，邮箱，编号，学历.... |

### 4-2 拆表的关键：外键

|  |
| --- |
| \* 外键约束作用：保证数据的正确性  \* 外键约束特点：外键约束也是约束的一种；不同于：非空/主键/唯一约束等，其约束源于其他的表，所以称为外键。    \* 结合案例：我们要保证，学生表中teacher\_id的取值，不能乱写，其取值的必须是教师表tid字段已经存在的值。  \* 即：幽灵不能当老师。    2. 语法  \* 添加外键:  alter table 表 add [CONSTRAINT 外键名] foreign key(字段) references 表(字段);  \* 删除外键  alter table 表名 drop foreign key 外键约束名  附：也可以在建表的时，一起指定外键：  create table teacher (  tid int primary key auto\_increment,  tname char(10)  );  create table student (  sid int primary key auto\_increment,  sname char(10),  teacher\_id int,  foreign key (teacher\_id) references teacher(tid)  );      3. 注意事项  a. 外键一定要指向主键(有主键才有外键)  b. 外键列的类型和长度,一定要和参照的主键的类型和约定长度一致。  c. 有外键约束后，删除表，必须先删除从表，后删除主表。  d. 外键的取值，必须来源于主键的已存在的值，但可以取null。 |

### 4-3 表与表的关系

部门与员工：

一个部门有多个员工 1个员工对应一个部门

部门表：部门编号（主键），部门名称

员工：员工编号（主键），员工姓名，员工所属的部门（外键表达）

1对多：在多的一方，新增一个字段，作为外键，指向1的一方的主键。

学生与课程

多对多：1个学生选择多门课程，1个课程，可以被多个学生选择

学生表：编号（主键），姓名

课程表：编号（主键），名称

中间表：

外键，指向学生表的主键

外键，指向课程表的主键。

？ 中间的主键怎么处理？

联合主键。

扩展：

用户 角色 权限表

用户角色表 角色权限表

丈夫与妻子：1对1

一种方案： 把任意一方，看做是多。看做特殊的1对多

第二种；合成一个表

### 4-4 扩展：约束的理解

4-4-1 约束是如何实现的

常见约束： 非空，唯一，主键，外键

数据库，有约束，而且有效果。 如何实现？

约束的实现依赖于数据库管理系统。

对于非空约束，管理系统，通过一个if的判断，如果是null值，直接报错，不操作数据库文件

对于唯一约束，管理系统，需要对表进行一个全表查询，查询当前值，是否在表中对应字段已经存在，如果不存在，可以正常操作数据库文件；如果存在，直接报错。

## 4-5 如何理解：表中无外键，心中有外键

在现在的开发中，所有的表的定义，现象是：没有外键约束。

老师表（老师编号，姓名，年龄）

学生表（学生编号，姓名，年龄， 所属老师的编号）

### 4-5 多表查询

#### 4-5-1 概念：笛卡尔积

定义

\* 笛卡尔积是一个数学概念。

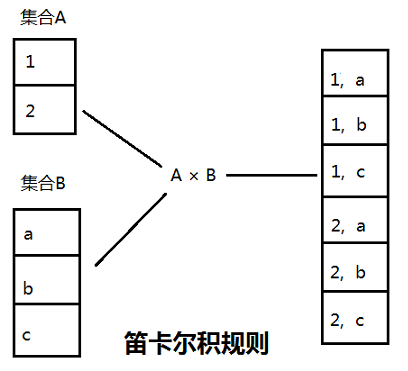
\* 两个集合X和Y的笛卡尓积，又称直积，表示为X × Y，第一个对象是X的成员而第二个对象是Y的所有可能有序对的其中一个成员

简单的案例

A={1,2}

B={a, b ,c}

A × B = {(1,a),(1,b),(1,c),(2,a),(2,b),(2,c)};



#### 4-5-2 查询的定义

交叉查询：笛卡尔积

显式内连接

隐式内连接

左外连接

右外连接：

什么情况下，用左外和右外？

查询的语句，特意指定，如果不存在也要显示的情况

|  |
| --- |
| 案例：  -- 新建表  create table department(did int primary key auto\_increment,dname char(10));  create table employee (eid int primary key auto\_increment, ename char(10),fk\_did int,  foreign key (fk\_did) references department(did));    -- 插入测试数据  insert into department values(null,"研发部"),(null,"营销部");  insert into employee values(null,"赵四",1),(null,"刘能",2),(null,"谢光坤",2);    -- 外连接准备  insert into department values(null,"总裁办");  insert into employee values(null,"谢大脚",null);  查询所有的员工及其归属部门的信息,某个员工，不归属于任意部门，也要显示。  select \* from department d right join employee e on d.did =e.fk\_did |

#### 4-5-3 多表查询入门

a. 查询的诀窍：

|  |
| --- |
| a. 先确定数据要用到哪些表  b. 将多个表先通过笛卡尔积变成一个表 【看作是一个虚拟表，数据更多，但存在大量的无效数据】  c. 然后去除不符合逻辑的数据 【提示：根据表与表的关系去掉】  d. 最后把“数据扩张之后的虚拟表”，看作单表，加上查询条件即可。 |

b. 实战

|  |
| --- |
| -- 新建表  create table department(did int primary key auto\_increment,dname char(10));  create table employee (eid int primary key auto\_increment, ename char(10),salary int,fk\_did int,  foreign key (fk\_did) references department(did));    -- 插入测试数据  insert into department values(null,"研发部");  insert into department values(null,"营销部");  insert into department values(null,"总裁办");      insert into employee values(null,"赵四",10000,1);  insert into employee values(null,"刘能",8000,1);  insert into employee values(null,"赵玉田",7000,1);  insert into employee values(null,"谢光坤",5000,2);  insert into employee values(null,"谢大脚",12000,2);  insert into employee values(null,"谢永强",10000,3);    . 查询案例  \* 技巧：**我的眼里只有你---始终看作是一个单表**（不管是真实的，还是虚拟的）  去掉，没有意义的数据之后，形成的虚拟表，看做是单表  a. 案例1：查询姓"谢"的员工信息    b. 案例2：查询姓"谢"的员工信息及其部门信息。    c . 案例3：查询各个部门姓"谢"的员工的员工数量。 |

#### 4-5-4 子查询

一定要记住，子查询的结果有3种情况及其处理方法：

1. 一行一列：看做是一个值。 使用“=”，“>””<”
2. 多行一列：看做是一个数组，使用in , not in
3. 多行多列：看做是一个虚拟表，放在from之后，使用

定义

\* 子查询就是嵌套查询，一条语句中存在2个及以上select，就是子查询。

\* 即：一个select语句中包含另一个的select语句。

实战

|  |
| --- |
| 1. 案例  a. 2个基本查询  \* 查询工资比10000高的员工名字  \* 查询刘能的工资    b. 扩展复杂查询  \* 查询工资比刘能高的员工名字。 |

纠结：老师应该用多表查询，还是子查询？

那种好用，用哪个

## 5 多表查询加强

### （单独写的讲义）

## 6 数据库深入

### 理解视图，索引，触发器，存储过程的价值

索引：单独讲

视图：

视图的本质：实际存储的是一个查询语句。没有真正的一个表。

视图的作用：a. 保护关键数据的隐私。 b. 可以简化复杂的sql查询。

触发器：

最大的价值，体现：自动执行

储存过程：

把多条sql语句合并在一起。

转账案例：

update account set total=total+500 where name=”张三”

update account set total=total-500 where name=”李四”

减少与数据库服务器的交互次数，节省系统响应时间。

缺点： 存储过程，不好编写，也不好调试。而且，因为数据库服务器特别贵，我们应该尽量节省数据库服务器的资源， 实际开发中个，很少使用存储过程。

## 7 事务

第一反应是：转账案例

事务是数据库的专用概念。

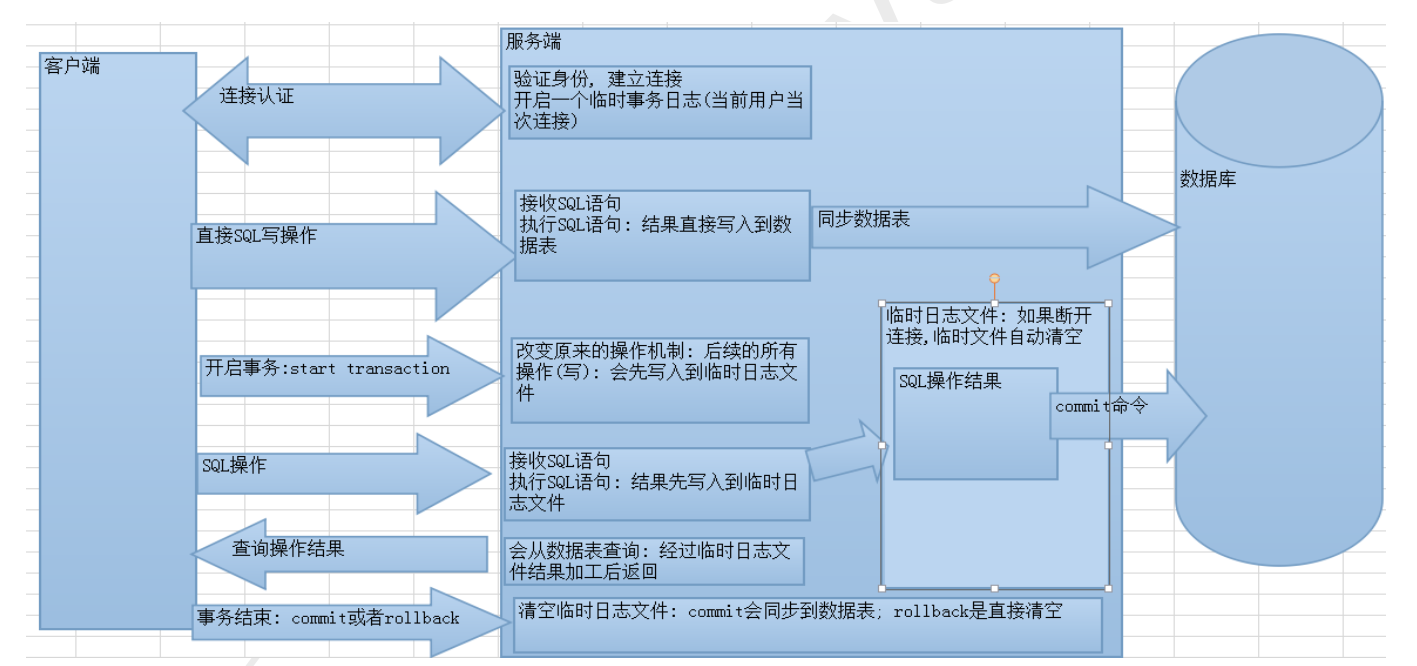
### 7-1 基本概念

事务的作用：保证数据的正确性。

### 7-2 事务的4个特性

|  |
| --- |
| 本篇讲诉数据库中事务的四大特性（ACID），并且将会详细地说明事务的隔离级别。  　　如果一个数据库声称支持事务的操作，那么该数据库必须要具备以下四个特性：  ⑴ 原子性（Atomicity）  　　原子性是指事务包含的所有操作要么全部成功，要么全部失败回滚，这和前面两篇博客介绍事务的功能是一样的概念，因此事务的操作如果成功就必须要完全应用到数据库，如果操作失败则不能对数据库有任何影响。  ⑵ 一致性（Consistency）  　　一致性是指事务必须使数据库从一个一致性状态变换到另一个一致性状态，也就是说一个事务执行之前和执行之后都必须处于一致性状态。  　　拿转账来说，假设用户A和用户B两者的钱加起来一共是5000，那么不管A和B之间如何转账，转几次账，事务结束后两个用户的钱相加起来应该还得是5000，这就是事务的一致性。  ⑶ 隔离性（Isolation）  　　隔离性是当多个用户并发访问数据库时，比如操作同一张表时，数据库为每一个用户开启的事务，不能被其他事务的操作所干扰，多个并发事务之间要相互隔离。  　　即要达到这么一种效果：对于任意两个并发的事务T1和T2，在事务T1看来，T2要么在T1开始之前就已经结束，要么在T1结束之后才开始，这样每个事务都感觉不到有其他事务在并发地执行。  　　关于事务的隔离性数据库提供了多种隔离级别，稍后会介绍到。  ⑷ 持久性（Durability）  　　持久性是指一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性的，即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。  　　例如我们在使用JDBC操作数据库时，在提交事务方法后，提示用户事务操作完成，当我们程序执行完成直到看到提示后，就可以认定事务以及正确提交，即使这时候数据库出现了问题，也必须要将我们的事务完全执行完成，否则就会造成我们看到提示事务处理完毕，但是数据库因为故障而没有执行事务的重大错误。 |

### 7-3 数据库事务的原理图



### 7-4感受事务的存在

从mysql和oracle的一个区别说起：

mysql默认自动提交， oracle默认不会自动提交

### 7-5 事务在实际开发中的体现

事务，在servie层处理。

在实际开发中，由spring负责处理事务操作，但是，我们配置切入点的时候，一定配置在service层上。

## 8 并发问题与隔离级别

### 8-1. 事务的并发问题

1． 介绍

a. 在实际的使用中，数据库被多个用户同时访问，即出现多个事务并发执行的情况。并且多个事务“同时使用相同的数据”时，可能出现并发问题。

b. 并发问题分类

(1) 脏读: 一个事务读到了另一个事务未提交的数据.

(2) 不可重复读: 一个事务读到了另一个事务已经提交的update数据,导致多次查询结果不一致.

(3) 虚读(幻读): 一个事务读到了另一个事务已经提交的insert数据,导致多次查询结构不一致.

2. 理解并发问题的前提

举例说明，什么情况下，出现多个事务“同时使用相同的数据”？

\* --?--

3. 事务的隔离级别

a. 解决事务并发问题，设计了多种隔离级别

b. 数据库事务的隔离级别有4个

(1). 由低到高依次为Read uncommitted（读未提交）、Read committed（读提交）、Repeatable read（重复读）、Serializable（序列化）

(2). 这四个级别可以逐个解决脏读、不可重复读、幻读这几类问题。

隔离级别 脏读 不可重复读 虚读(幻读)

Read uncommitted × × ×

Read committed √ × ×

Repeatable read √ √ ×

Serializable √ √ √

(备注： √: 可能解决 ×: 不能解决)

### 8-2. 小故事：记忆并发问题和隔离级别

a.隔离级别:读未提交(Read uncommitted)

(1) 领导把5000元工资打到长贵的账号上，该事务未提交，长贵去查看账户，就看到了5000块工资。

(2) 上述现象即“脏读”。

脏读：两个并发的事务，“事务A：领导给长贵发工资”、“事务B：长贵查询工资账户”，事务B读取了事务A尚未提交的数据。

(3) 当隔离级别设置为Read uncommitted(读未提交)时，就可能出现脏读，避免脏读，请看下一个。

b. 隔离级别:读提交(Read committed)

(1) 发工资后，长贵去happy，读到卡里有2000元，同时他老婆谢大脚网上转账，把长贵工资卡的2000元转走，并在长贵之前提交了事务。当长贵扣款时，卡已经没有钱，扣款失败，长贵十分纳闷，钱到那里去了...

(2) 上述情况即“不可重复读”。

不可重复读：两个并发的事务，“事务A：长贵消费”、“事务B：谢大脚网上转账”，“事务A先读取数据，事务B紧接更新数据，并提交事务，而事务A再次读取该数据时，数据已经发生了改变”。

(3) 当隔离级别设置为Read committed(读提交)时，解决了脏读，不能解决不可重复读。大多数数据库的默认级别就是Read committed，比如Sql Server , Oracle。解决不可重复读，请看下一个。

c. 隔离级别：重复读(Recpeatable read)

(1) 长贵拿着工资卡去消费时，一旦开始读取工资卡信息（即事务开始），谢大脚就不可能对该记录进行修改（转账）。但谢大脚可以查看长贵工资卡的消费记录。有一天，她正在查询到长贵当月的总消费金额（select sum(amount) from transaction where month = 本月）为80元，而长贵正好在外面happy后买单1000块，并提交了事务。谢大脚把当月消费的明细打印时，发现消费总额为1080元，其很诧异，多了一行消费1000块的记录，以为出现了幻觉，即“幻读”。

(2) 幻读:两个并发的事务，"事务A读取某一范围的数据行时（不是同一行数据）"， “事务B在该范围内插入了新行”,"事务A读取该范围的数据行时，会发现有新的“幻影” 行

(3) 当隔离级别设置为Recpeatable read(重复读)时，解决了脏读和不可重复读，不能解决幻读。解决不可重复读，请看下一个。

d. 隔离级别：序列化(Serializable)

\* 最严格的事务隔离。要求事务序列化执行，即一个接一个执行，不能并发执行。所以，不会存在任何并发问题。但效率低，很少使用。

### 8-3 如何处理

使用数据库系统默认的隔离级别即可。

mysql默认：重复读 oracle默认：读提交

## 9 数据结构入门

### （单独写的讲义）

## 10 数据库优化及索引深入

### （单独写的讲义）