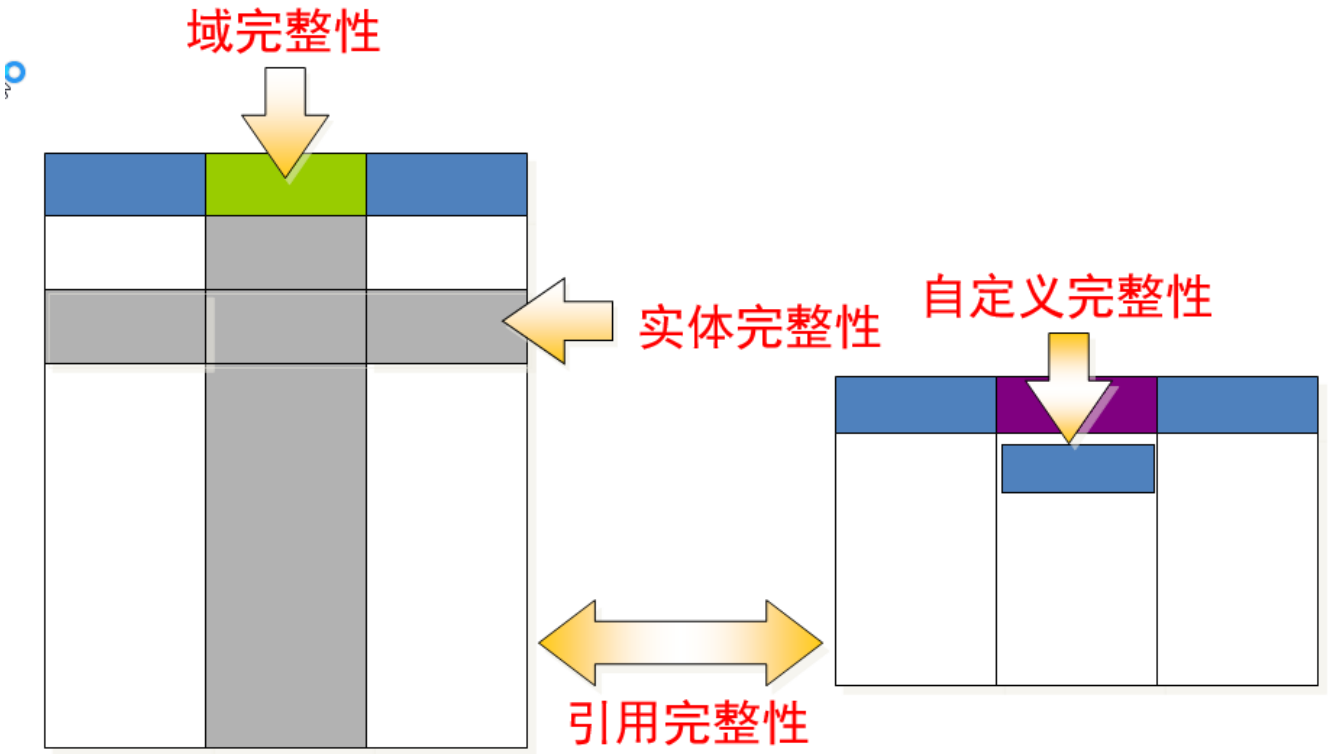
数据库中实体的关系

一、数据的完整性和事务



1.实体的完整性,一条记录,就是一个实体,如果记录无法区分,则失去了实体的完整性
2.域完整性:如果有两个字段无法区分,则失去了域完整性
3.引用的完整性:两个表的对应记录不完整,则失去了引用完整性
4.自定义完整性:自己定义的一套规则

(1).保证实体的完整

```
1.主键的约束(primary key)
2.自动增长的列(auto_increment)
3.唯一键(unique)
```

(2).保证域的完整性

```
1.数据类型的约束
2.默认值(default)
3.非空约束(not null)
```

(3).引用的完整性

```
应用外键(foreign key)
```

(4).自定义完整性

```
1.存储过程(相当于python中的自定义函数)
2.触发器
```

二、外键

外键:从表的公共字段

外键的约束主要是用来保证引用的完整性的,主外键的名字可以不一样,但是数据类型可以一样.

```
#特点
1.主表中不存在的记录,从表中不能插入
2.从表已存在的记录,主表中不能删除
3.先删除从表,再删除主表
```

```
#两种串联的操作:
1.set null: 让一个字段设置为NULL
2.cascade : 跟着主表的变化而变化

#添加外键
alter table score add foreign key(id) references
student(id) on delete cascade on update cascade;
#添加外键,并指定外键的名称
```

```
alter table score add CONSTRAINT `score_id` FOREIGN KEY
(`id`) REFERENCES `student` (`id`) ON DELETE CASCADE ON
UPDATE CASCADE;

#删除外键
#score_ibfk_1 外键的名字,外键可以有多个
alter table score drop foreign key score_id;

#外键只能在innodb的引擎上使用
```

三、实体之间的关系

实体的关系:

```
1.一对一
2.一对多
3.多对一
4.多对多
```

(1).一对一:主键关系

stuinfo

stuno(学号)	name(姓名)
1	王健林
2	许家印

stuno(期末考试)	score
1	100
2	120

(2)一对多|多对一

user

uid	account	pwd	email
1	admin	123456	123@qq.com
2	root	123456	126@126.com

order(订单)

oid(订单的编号)	uid	gid	create_at
201904037000001	1	7000	2019/04/03
201904047000001	1	7000	2019/04/04

(3)多对多

user

uid	account	pwd	email
1	admin	123456	123@qq.com
2	root	123456	126@126.com

address

address	uid	mobile	name
上海徐汇区1208弄A小区3栋1608	1	13555555555	马云
上海徐汇区1208弄A小区3栋1608	1	13666666666	雷军
上海徐汇区1208弄B小区3栋1608	1	13777777777	吴军
上海徐汇区1208弄B小区3栋1608	1	13888888888	李斌
上海徐汇区1208弄A小区3栋1608	2	13555555555	马云
上海徐汇区1208弄A小区3栋1608	2	13666666666	雷军

四、数据库的设计

公司要做一个项目，首先项目管理获得需求，知道项目是什么类型的，然后产品经理负责产品的规划，设计原型

UI将需求的草图给UI，UI可以绘制E-R图，或者是DB自己构建E-R图

DB自己根据E-R图设计数据库，建立表，设定关联度.

码农看到E-R图可以干嘛，我们根据E-R图上的需求写代码

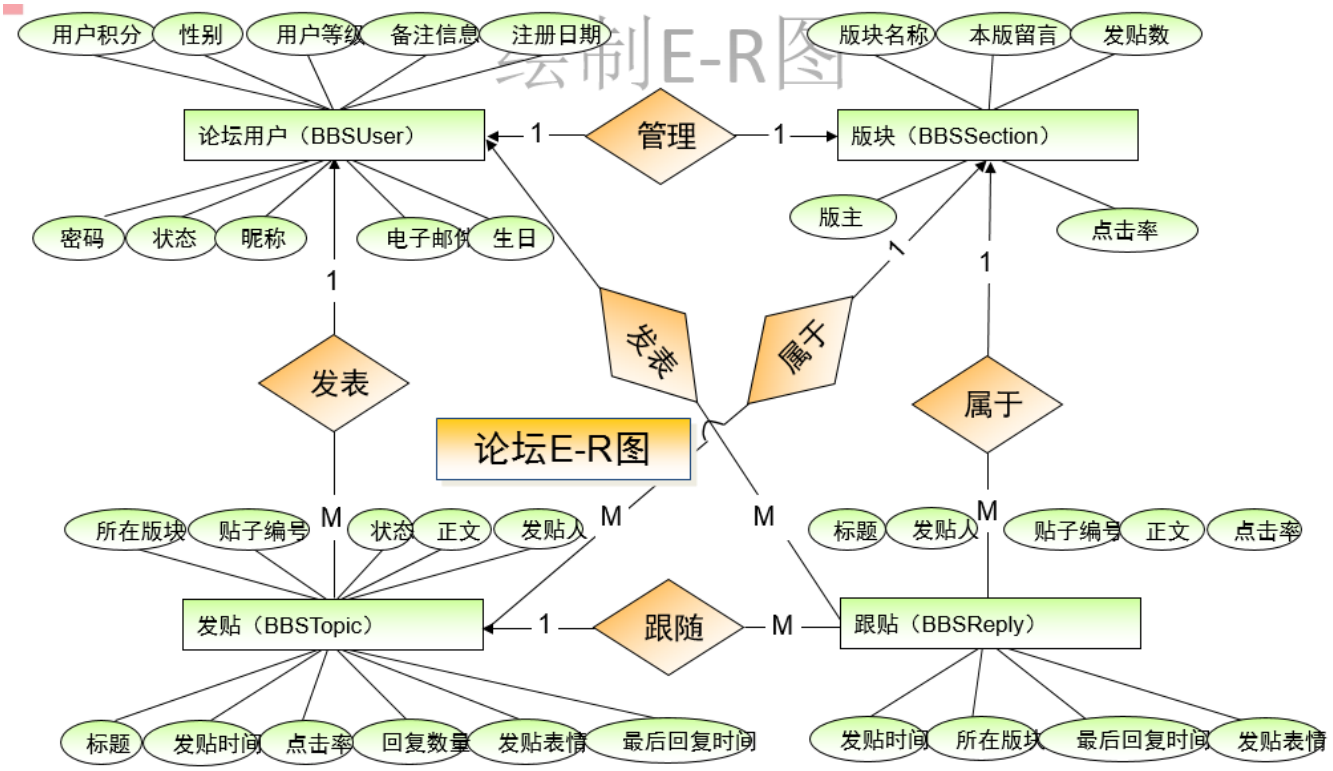
论坛用户：	主贴	回贴	版块
• 呢称 • 密码 • 电子邮件 • 生日 • 性别 • 用户的等级 • 备注信息 • 注册日期 • 状态 • 积分	• 发贴人 • 发贴表情 • 回复数量 • 标题 • 正文 • 发贴时间 • 点击数 • 状态： • 最后回复时间	• 贴子编号 • 回贴人， • 回贴表情 • 标题 • 正文 • 回贴时间 • 点击数	• 版块名称 • 版主 • 本版格言 • 点击率 • 发贴数

- E-R图

E-R图是描述实体和实体之间的关系的

语法：
1.矩形代表实体
2.椭圆形代表实体拥有的属性
3.菱形代表实体之间的关系

博客的E-R图:



#用户和板块之间的关系

1.某个用户是版主,版主管理板块

2.普通用户和版块之间没有直接的关系,用户发帖或者用户评论间接的和版块之间形成关系

#用户和帖子之间的关系

1.用户发表了帖子

2.用户评论了某个帖子

#用户和评论之间的关系

1.用户发表了评论

2.用户发表了帖子,被其他人评论了

3.如果有二级评论,你的评论被人喷了

#帖子和版块之间的关系

帖子属于版块

五、数据的规范

(1)第一范式

第一范式:确保每一列原子化(不可分割)

BuyerID	Address
1	中国北京市
2	美国纽约市
3	英国利物浦
4	日本东京市
…	…

BuyerID	Country	City
1	中国	北京
1	中国	北京
4	日本	东京
2	美国	纽约
…	…	…

(2)第二范式

第二范式:,基于第一范式,一张表只能描述一件事情,非主键字段必须依赖主键字段(不论在什么情况下主键都是唯一的)

Orders

字　段	例　子
订单编号	001
产品编号	A001
订购日期	2000-2-3
价　格	$29.00
…	…

Orders

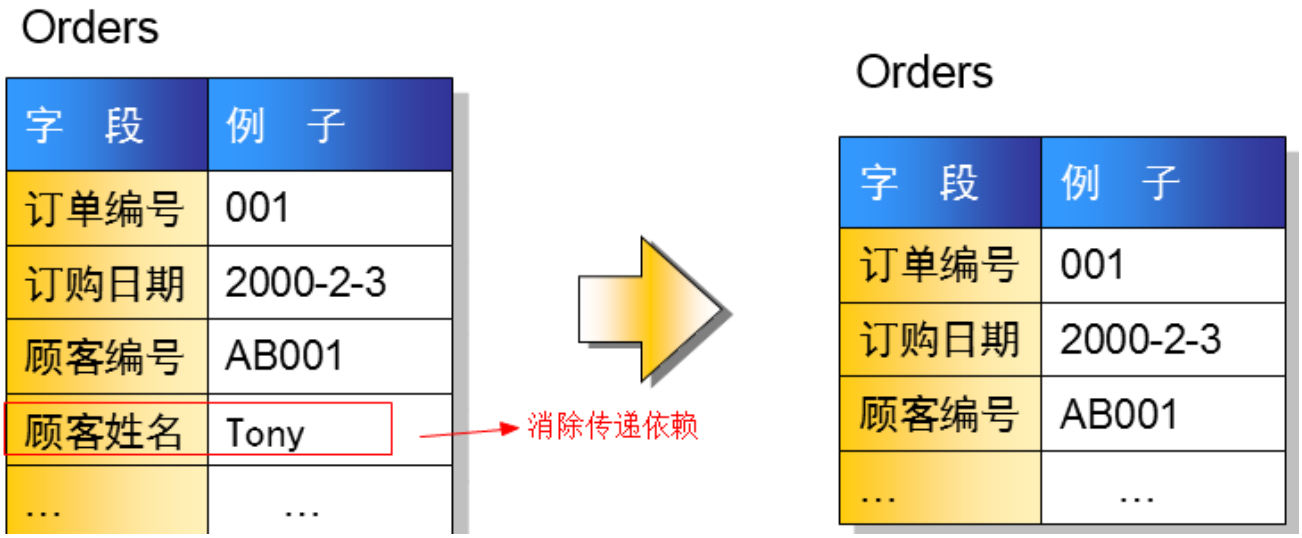
字　段	例　子
订单编号	001
订购日期	2000-2-3

Products

字　段	例　子
产品编号	A001
价　格	$29.00

(3)第三范式

第三范式:基于第二范式,消除传递依赖(一个主键字段可以确定其它的信息)



六、规范化和性能

高考成绩查询系统:高并发

不符合三范式

stuno(考号)	姓名	语文	数学	总分
1	小明	130	120	250

select * from gaokao where stuno=1;

规范化:

stuno(考号)	姓名
1	小明
2	小强

stuno(考号)	语文	数学
1	130	120

select *,ch+math as score from A left join B on A.stuno = B.stuno where A.stuno = 1;

总结:`性能的完备良好的时候,选择规范化;性能不足,优先考虑性能`

七、事务

1. 事务简介

事务主要用于处理操作量大、复杂度高、并且关联性强的数据。

比如说, 在人员管理系统中, 你删除一个人员, 你即需要删除人员的基本资料, 也要删除和该人员相关的信息, 如信箱, 文章等等, 这样, 这些数据库操作语句就构成一个事务!

在 MySQL 中只有 Innodb 存储引擎支持事务。

事务处理可以用来维护数据库的完整性, 保证成批的 SQL 语句要么全部执行, 要么全部不执行。主要针对 insert, update, delete 语句而设置

2. 事务四大特性

在写入或更新资料的过程中, 为保证事务 (transaction) 是正确可靠的, 所必须具备的四个特性 (ACID):

1. 原子性 (Atomicity):
    - 事务中的所有操作, 要么全部完成, 要么全部不完成, 不会结束在中间某个环节。
    - 事务在执行过程中发生错误, 会被回滚 (Rollback) 到事务开始前的状态, 就像这个事务从来没有执行过一样。

2. 一致性 (Consistency)：

   在事务开始之前和事务结束以后, 数据库的完整性没有被破坏。 这表示写入的资料必须完全符合所有的预设规则, 这包含资料的精确度、串联性以及后续数据库可以自发性地完成预定的工作。

3. 隔离性 (Isolation)：

   数据库允许多个并发事务同时对其数据进行读写和修改的能力, 隔离性可以防止多个事务并发执行时由于交叉执行而导致数据的不一致。 事务隔离分为不同级别, 包括:

   1. 读取未提交 (Read uncommitted)
      - 所有事务都可以看到其他未提交事务的执行结果
      - 本隔离级别很少用于实际应用，因为它的性能也不比其他级别好多少
      - 该级别引发的问题是——脏读(Dirty Read)：读取到了未提交的数据
   2. 读提交 (read committed)
      - 这是大多数数据库系统的默认隔离级别（但不是MySQL默认的）
      - 它满足了隔离的简单定义：一个事务只能看见已经提交事务做的改变
      - 这种隔离级别出现的问题是: 不可重复读(Nonrepeatable Read)：

        不可重复读意味着我们在同一个事务中执行完全相同的 select 语句时可能看到不一样的结果。

        导致这种情况的原因可能有：

        - 有一个交叉的事务有新的commit，导致了数据的改变;
        - 一个数据库被多个实例操作时,同一事务的其他实例在该实例处理其间可能会有新的commit
   3. 可重复读 (repeatable read)

    - 这是MySQL的默认事务隔离级别
    - 它确保同一事务的多个实例在并发读取数据时，会看到同样的数据行
    - 此级别可能出现的问题: 幻读(Phantom Read)：当用户读取某一范围的数据行时，另一个事务又在该范围内插入了新行，当用户再读取该范围的数据行时，会发现有新的“幻影” 行
    - InnoDB 通过多版本并发控制 (MVCC，Multiversion Concurrency Control) 机制解决幻读问题；
    - InnoDB 还通过间隙锁解决幻读问题
4. 串行化 (Serializable)
    - 这是最高的隔离级别
    - 它通过强制事务排序，使之不可能相互冲突，从而解决幻读问题。简言之,它是在每个读的数据行上加上共享锁。MySQL锁总结
    - 在这个级别，可能导致大量的超时现象和锁竞争

4. 持久性 (Durability)：

   事务处理结束后, 对数据的修改就是永久的, 即便系统故障也不会丢失。

3. 语法与使用

- 开启事务: `BEGIN`或`START TRANSACTION`
- 提交事务: `COMMIT`, 提交会让所有修改生效
- 回滚: `ROLLBACK`, 撤销正在进行的所有未提交的修改
- 创建保存点: `SAVEPOINT identifier`
- 删除保存点: `RELEASE SAVEPOINT identifier`
- 把事务回滚到保存点: `ROLLBACK TO identifier`
- 设置事务的隔离级别: `SET TRANSACTION`

  InnoDB 提供的隔离级别有

- `READ`
- `UNCOMMITTED`
- `READ COMMITTED`
- `REPEATABLE READ`
- `SERIALIZABLE`

4. 示例

```
create table `abc` (
    id int unsigned primary key auto_increment,
    name varchar(32) unique,
    age int unsigned
) charset=utf8;

begin;
insert into abc (name, age) values ('aa', 11);
insert into abc (name, age) values ('bb', 22);
-- 在事务中查看一下数据
-- 同时另开一个窗口，连接到 MySQL 查看一下数据是否一样
select * from abc;
commit;

begin;
insert into abc (name, age) values ('cc', 33);
insert into abc (name, age) values ('dd', 44);
update abc set age=77 where name='aa';
-- 在事务中查看一下数据
select * from abc;
rollback;

select * from abc;  -- 事务结束后在查看一下数据
```