数据建模: 范式和反范式

> **Organization**: 千锋教育 Python 教学部
> **Date** : 2019-10-17
> **Author**: 张旭

范式是数据库规范化的一个手段，是数据库设计中的一系列原理和技术，用于减少数据库中的数据冗余，并增进数据的一致性。

数据规范化通常是将大表分成较小的表，并且定义它们之间的关系。这样做的目的是为了避免冗余存放数据，并确保数据的一致性。添加、删除和修改数据等操作可能需要修改多个表，但只需要修改一个地方即可保证所有表中相关数据的一致性。由于数据分布在多个表之间，因此检索信息可能需要根据表之间的关系联合查询多个表。

数据规范化的实质是简单写、复杂读。写入操作比较简单，对于不同的信息，分别修改不同的表即可;而读取数据则相对复杂，检索数据的时候，可能需要编写复杂的SQL来联合查询多个表。

第一范式 (1NF)

第一范式是指数据库表的每一列(属性)都是不可分割的基本数据项，这就要求数据库的 **每一列都只能存放单一值**，即实体中的某个属性不能有多个值或不能有重复的属性。

第一范式是对关系模式的基本要求。

关键点: 每一列都只能存放单一值

例如我们开发微博时的 User 表和微博表，一个用户可以发表多个微博，但设计时需要将用户数据和微博数据单独存放。

id	name	age	weibo
1	tom	21	"今天天气不错,2019-09-30", "我们下午没课, 2019-10-22"
2	bob	26	"睡毛觉,2019-09-30", "起来嗨, 2019-10-22"
3	lucy	23	"我想吃火锅,2019-09-30", "海底捞也可以, 2019-10-22"

第二范式 (2NF)

第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，一个数据表符合第二范式的前提是该数据表符合第一范式。
它的规则是要求数据表里的所有数据都要和该数据表的主键有完全相依的关系; 如果有哪些数据只和主键的一部分有关的话，就得把它们独立出来变成另一个数据表。

关键点: 非主键字段完全依赖于主键

错误范例:

一个微博表，content 字段依赖 wb_id，而不依赖主键 user_id

user_id	name	age	wb_id	content	date
1	tom	22	5	今天天气不错	2019-09-30
1	tom	21	6	我们下午没课	2019-10-22
2	bob	26	8	睡毛觉	2019-09-30
2	bob	26	8	起来嗨	2019-10-22
3	lucy	23	9	我想吃火锅	2019-09-30
3	lucy	23	10	海底捞也可以	2019-10-22

第三范式 (3NF)

第三范式要求所有非主键属性都只和主键有相关性，而非主键属性之间应该是独立无关的。

更通俗一点讲，非主键列必须直接依赖于主键，不能存在传递依赖。即不能存在: 非主键列 A 依赖于非主键列 B，非主键列 B 依赖于主键的情况。

第三范式（3NF）是第二范式（2NF）的一个子集，即满足第三范式（3NF）必须满足第二范式（2NF）。

关键点: 消除传递依赖 (非主字段不依赖于其它非主字段)

错误范例：

一个订单表，总价字段 total 依赖另外两个字段，所以应该去掉。

order_id	user	uint_price	num	total
15	bob	8	3	24
16	tom	5	6	30
17	lucy	7	8	56

反范式

反范式是试图通过增加冗余数据或通过分组数据来优化数据库读取性能的过程。在某些情况下，反范式是解决数据库性能和可伸缩性的极佳策略。

范式化的设计是在不同的有关系的表中存储不同的信息，如果需要查询信息往往需要连接多个表，如果连接的表很多，将会导致很多随机I/O，那么查询可能会非常慢。

一种做法是反范式的数据表设计。由于多了冗余数据，因此数据的一致性需要靠数据库约束或应用程序来保证。传统商业数据库一般通过施加数据库约束来确保数据的一致性，而互联网数据库一般靠应用程序来确保数据的一致性。

例如，在微博表中，除了增加作者 id 外，还可以把作者姓名同时加进来。