







(三)MySQL之库表设计篇:一、二、三、四、五范式、BC范式与反范式详解!



竹子爱熊猫 (17.5) 2022年09月23日 15:32 · 阅读 4452

✓ 已关注

罚 引言

66

本文为掘金社区首发签约文章,14天内禁止转载,14天后未获授权禁止转载,侵权必究!

99

MySQL 的库表设计,在很多时候我们都是率性而为,往往在前期的设计中考虑并不全面,同时对于库表结构的划分也并不明确,所以很多时候在开发过程中,代码敲着敲着会去重构某张表结构,甚至大面积重构多张表结构,这种随心所欲的设计方式,无疑给开发造成了很大困扰。



<u>1</u> 5

29









- ①数据库三大范式 (1NF、2NF、3NF)
- ③第四范式(4NF) 和第五范式: 完美范式 (5NF)
- ②巴斯-科德范式 (BCNF)
- ④反范式设计

不过对于上述的几种设计范式,大部分小伙伴应该仅了解过三范式,对于其他的应该未曾接触,那在本篇中会重点阐述库表设计时,会用到的这些范式。

一、数据库三大范式

范式(Normal Form)在前面也提到过,它就是指设计数据库时要遵守的一些原则,而数据库的三大范式,相信诸位在学习数据库知识时也定然接触过。三大范式之间,它们是递进的关系,也就是后续的范式都基于前一个范式的基础上推行,就好比下面这句话:

66

今天我要先炒菜,然后吃饭,最后洗碗。

9

炒菜、吃饭、洗碗三者也属于递进关系,后者都建立在前者之上,其顺序不能颠倒,比如先吃饭再炒菜,这必然是行不通的。数据库的三大范式也一样,第二范式必须建立在第一范式的基础之上,如若设计的库表第一范式都不满足,那定然是无法满足第二范式的。

"

写在前面的话:其实对于数据库三范式相关的资料,网上也有很多很多,但大部分资料都涉及了太多的概念,通篇看下来也很难让人理解,因此下述的三范式则会结合具体的设计实例来让诸位彻底理解三范式。

99

∠ 1.1、第一范式 (1NF) (∠

















在上述的学生表中,其中有一个 student 学生列,这一列存储的数据则明显不符合第一范式:原子性的规定,因为这一列的数据还可以再拆分为姓名、性别、身高三项数据,因此为了符合第一范式,应该将表结构更改为:



将 student 这一列数据,分别拆分为姓名、性别、身高三列,然后分别存储对应的数据才合理,通过这样的优化后,此时 zz_student 这张表则符合了数据库设计的第一范式。

66

那此刻思考一下:如果不去拆分列满足第一范式,会造成什么影响呢?

99

。旁方些语言和主力问工法仍仍仍开出面的子艺

<u>6</u>2 53

29









简单来说,如果按照原本那张形式去做业务开发,显然操作起来会更加麻烦且复杂一些,但第一范式的原子性,除开对列级别生效之外,行级别的数据也是同理,也就是每一行数据之间是 互不影响的,都是独立的一个整体。

∠ 1.2、第二范式 (2NF)



上述的第一范式还是比较容易理解,紧接着来看看第二范式,第二范式的要求表中的所有列,其数据都必须依赖于主键,也就是一张表只存储同一类型的数据,不能有任何一列数据与主键没有关系,还是上面的那张表数据为例:



虽然此时已经满足了数据库的第一范式,但此刻观察 course 课程、 score 分数这两列数据,跟前面的几列数据实际上依赖关系并不大,同时也由于这样的结构,导致前面几列的数据出现了大量冗余,所以此时可以再次拆分一下表结构:

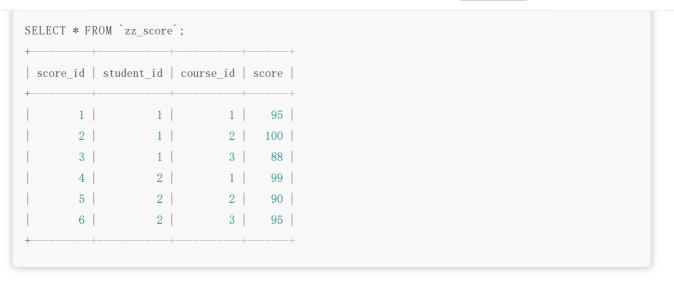












经过上述结构优化后,之前的一张表此时被我们拆分成学生表、课程表、成绩表三张,每张表中的 id 字段作为主键,其他字段都依赖这个主键。无论在那张表中,都可以通过 id 主键确定其他字段的信息。

66

主键可以不用 id , 但最好是自增的主键 ID , 这跟索引有关, 后续索引篇详细讲解。

9

此时再将目光看到先后两张学生表,原本的学生表有六条学生记录,其中有四条是冗余数据,此时的学生表则只有两条数据,同时这张学生表中只存储学生信息相关的数据。经过本次结构优化后,每张表的业务属性都具备"唯一性",也就是每张表都只会描述了"一件事情",不会存在一张表中会出现两个业务属性(例如之前的学生表包含了学生信息和课程成绩)。

∠ 1.3、第三范式 (3NF)



前面已经对第一范式、第二范式进行了直观阐述,接下来聊一聊数据库的第三范式,第三范式要求表中每一列数据不能与主键之外的字段有直接关系,怎么理解呢?基于上述的例子:











department 表示当前学生所属的院校, dean 则表示这个院系的院长是谁。一般来说,一个学生的院长是谁,首先是取决于学生所在的院系的,因此最后的 dean 字段明显与 department 字段存在依赖关系,因此需要进一步调整表结构:



经过进一步的结构优化后,又将原本的学生表拆为了院系表、学生表两张,学生表中则是只存储一个院系 ID ,由院系表存储院系相关的所有数据。至此,学生表中的每个非主键字段与其他非主键字段之间,都是相互独立的,之间不会再存在任何依赖性,所有的字段都依赖于主键。

那这里为什么要调整呢?不调整不行吗?还真不行,来简单思考一下不调整结构的情况下会发生什么问题:

- ①当一个院系的院长换人后,需要同时修改学生表中的多条数据。
- ②当一个院长离职后,需要删除该院长的记录,会同时删除多条学生信息。
- •

也就是如果设计的表结构,无法满足第三范式,在操作表时就会出现异常,使得整个表较难维护。

∠ 1.4、数据库三范式小结



到过田部口经挖床事设计的二节式做了青亚阁法 首结加下:















• 第三范式: 确保独立性, 表中除主键外, 每个字段之间不存在任何依赖, 都是独立的。

经过三范式的示例后,数据库中的表数量也逐渐多了起来,似乎设计符合三范式的库表结构, 反而更加麻烦了对吗?答案并非如此,因为在没有按照范式设计时,会存在几个问题:

- ①整张表数据比较冗余,同一个学生信息会出现多条。
- ②表结构特别臃肿,不易于操作,要新增一个学生信息时,需添加大量数据。
- ③需要更新其他业务属性的数据时,比如院系院长换人了,需要修改所有学生的记录。

但按照三范式将表结构拆开后,假设要新增一条学生数据,就只需要插入学生相关的信息即可,同时如果某个院系的院长换人了,只需要修改院系表中的院长就行,学生表中的数据无需发生任何更改。

66

因此,经过三范式的设计优化后,整个库中的所有表结构,会显得更为优雅,灵活性也会更强。

99

🖫 二、巴斯-科德范式与第四、五范式

第一阶段中,简单了解了库表设计时最基本的三大范式,但除此之外还有另外三种设计范式,即巴斯-科德范式与第四、第五范式,这后续三种范式可能有很多小伙伴没接触过,但当你尝试从网上去了解时,相信绝大部分能看到的资料你都看不懂,例如:

₽BCNF

观察上图中的描述,这一眼望过去几乎不是给人看的(没有诋毁的意思,单纯感慨~),其中涉及的码、完全函数依赖等名词,至少刚接触的小白是读不懂的,因此接下来则依旧采用上面那种案例+大白话的模式,简单阐述一下这三种设计范式。

∠ 2.1、巴斯-科德范式 (BCNF)



在了解后续这些范式之前。首先得弄明白一个概念。一般在一张表中。可以用干区分每行数















巴斯-科德范式也被称为 3.5NF ,至于为何不称为第四范式,这主要是由于它是第三范式的补充版,第三范式的要求是:任何非主键字段不能与其他非主键字段间存在依赖关系,也就是要求每个非主键字段之间要具备独立性。而巴斯-科德范式在第三范式的基础上,进一步要求:「任何主属性不能对其他主键子集存在依赖」。

66

对于上述的范式定义大家估计有些晕,那用大白话说简单一点,也就是规定了联合主键中的某列值,不能与联合主键中的其他列存在依赖关系,相信这样讲大家更加容易理解。当然,还是结合一个案例阐述。

9

先来看一张表:



例如这张学生表,此时假设以 classes 班级字段、 class_adviser 班主任字段、 name 学生姓名字段,组合成一个联合主键,在这里我们可以通过联合主键,确定学生表中任何一个学生的信息,比如:

66

熊竹老师管的计算机-2201班,哪个竹子同学有多高啊? 对于这个问题,可以通过上述的联合主键精准定位到表中第一条数据,并且最终能够给出答案为 185cm。

9

当然,在这里有小伙伴有疑惑,为什么这三个字段可以组成联合主键,和其他字段,例如身

<u>6</u>2 53

29









66

熊竹老师管的计算机-2201班,哪个男同学有多高啊?

99

现在以这个联合主键还能精准定位到每一条数据吗?不行的,因为表中有两位男同学,所以会扫描到多条数据,最终会得到 185cm、180cm 两个答案,显然「班级、班主任、性别」这三个字段具备重复性,不适合作为主键。

到这里,咱们分析一下,假设以「班级、班主任、学生姓名」三个字段组成联合主键,当前这张表是否符合前面的三大范式呢?

- 第一范式: 表中每列数据都不可再分, 具备原子性, 满足。
- 第二范式:表中每行数据都仅描述了学生信息这一种业务属性,具备唯一性,满足。
- 第三范式:除主键外,表中非主键字段之间都不存在依赖关系,具备独立性,满足。

经过上述分析后,当前这张表也符合前面聊到的三大范式,但没有问题了吗?有的,在这张表中,一条学生信息中的班主任,取决于学生所在的班级,比如「竹子同学、子竹同学」在「计算机-2201班」,所以它们的班主任都是「熊竹老师」,因此班主任字段其实也依赖于班级字段。那会造成什么问题呢?

- ①当一个班级的班主任老师换人后,需要同时修改学生表中的多条数据。
- ②当一个班主任老师离职后,需要删除该老师的记录,会同时删除多条学生信息。
- ③想要增加一个班级时,同时必须添加学生姓名数据,因为主键不允许为空。

通过上述分析可以明显得知,如果联合主键中的一个字段依赖于另一个字段,同样也会造成不小的问题,使得整张表的维护性变差,因此这里需要进一步调整结构:











经过结构调整后,原本的学生表则又被拆为了班级表、学生表两张,在学生表中只存储班级 ID,然后使用 classes id 班级 ID 和 name 学生姓名两个字段作为联合主键。

66

实际情况中,学生表应该有学生 ID 字段作为主键,因为同一个班级中也有可能会出现重名的现象,但这里是为了举例说明,不要纠结细节~

99

此时经过调整后,目前的学生表也满足了巴斯-科德范式,同时对于前面列出的三个问题,调整结构后也不复存在,比如换班主任后只需要更改班级表,无需修改学生表中的学生信息;增加班级时,只需要在班级表中新增数据,也不会影响学生表。

66

在这里更专业的做法,应该是对于班级表中的班主任老师信息,再进一步抽象出一张教师表。毕竟班主任字段还依旧与班级字段存在依赖关系,但班级表中的主键却是班级 ID,所以非主键字段之间存在关联,是不满足第三范式的(但这里大家清楚就好啦,我就不做了!)。

99

OK,经过上述一个案例的剖析后,大家对巴斯-科德范式也有了全面的认知,至于它为何被叫做 3.5 范式,相信大家也能够想清楚答案,因为巴斯-科德范式并没有定义新的设计规范,仅是对第三范式的做了补充及完善,修正了第三范式。

66

第三范式只要求非主键字段之间,不能存在依赖关系,但没要求联合主键中的字段不能存在依赖,因此第三范式并未考虑完善,巴斯-科德范式修正的就是这点。

90

1 53

29

√ 收藏









认识了巴斯-科德范式后,再来看看数据库的第四范式,第四范式是基于 BC 范式之上的,但在理解第四范式之前,首先得理解"多值依赖"的概念,先贴一下学术论文中常见的定义:

多值依赖



论文来源于: <u>《道客巴巴-多值依赖》</u>, 大部分网上资料的描述也都来自于这些学术论文。

99

能看明白嘛?看不明白就对了,对于这种概念看起来确实令人头大,没有相关的技术知识储备,就算挠破头皮也看不懂这段描述,因此简单说一下什么叫做多值依赖:

66

一个表中至少需要有三个独立的字段才会出现多值依赖问题,多值依赖是指表中的字段之间存在一对多的关系,也就是一个字段的具体值会由多个字段来决定。

77

这样写出来似乎比前面好理解一些了,但相对来说还是很绕,那就再上个例子:



上述是一个经典的业务,也就是一张用户角色权限表,先简单介绍一下表中各字段的信息:

53 كا

29

☆ 收藏









。 USER : 普通用户角色。

○ ADMIN:管理员角色。

。 ROOT : 超级管理员角色。

• permission 字段 -- 权限信息:

。 *: 超级管理员拥有的权限级别, *表示所有。

。 BACKSTAGE: 管理员拥有的权限级别,表示可以操作后台。

。 LOGIN: 普通用户拥有的权限级别,表示可以登录访问平台。

理解各字段的值后,假设以「用户名、角色、权限」三个字段作为联合主键,先来分析一下这 张表是否满足之前的范式:

- 表中每列数据都不可再分, 具备原子性, 满足第一范式。
- 表中数据都仅描述了用户权限这一种业务属性,具备唯一性,满足第二范式。
- 除主键外, 表中其他字段不存在依赖关系, 具备独立性, 满足第三范式。
- 联合主键中的用户、角色、权限都为独立字段,不存在依赖性,满足 BC 范式。

因为表中除开联合主键外,就剩下了一个性别字段,因此非主键字段必然是独立的,所以满足第三范式,但对于 BC 范式仅是勉强满足,因为「用户、角色、权限」之间存在一些依赖关系,不过这里先不管,毕竟是举例说明,因此假设是满足 BC 范式。

66

上述的表结构介绍清楚后,现在来聊聊什么叫做多值依赖。

9

此时假设我们需要新增一条数据,那表中的权限字段究竟填什么?这个值是需要依赖多个字段决定的,权限来自于角色,而角色则来自于用户。也就是说,一个用户可以拥有多个角色,同时一个角色可以拥有多个权限,所以此时咱们无法单独根据用户名去确定权限值,权限值必须依赖用户、角色两个字段来决定,这种一个字段的值取决于多个字段才能确定的情况,就被称为多值依赖。

"

到这里是不是就理解了多值依赖?再举个例子,也就是网上经典的例子。

90

53

29









		+
语文 计算机一班 竹熊老师	人教版-新课标教材	
语文 计算机二班 黑竹老师	人教版-现行教材	
语文 计算机三班 竹熊老师	北师大版教材	
数学 计算机一班 熊竹老师	人教版-新课标教材	
英语 计算机一班 黑熊老师	人教版-新课标教材	

上述是一张教师排课表,分别有课程、班级、老师、教材四个字段,一个课程会有多位老师授课,同时一个课程也会有多个版本的教材,此时就无法只根据课程、班级、老师任一字段决定教材字段的值,而是要结合班级、课程、老师三个字段,才能确定教材字段的值,比如计算机一班的语文课程,竹熊老师来上,用的是人教版-新课标教材,因此目前教材字段也存在多值依赖的问题,依赖于班级、课程、老师三个字段。

66

再经过一个案例的熏陶后,是不是对多值依赖的概念理解更深刻啦~

9

到这里为止,多值依赖的概念就讲清楚了,也正是由于多值依赖的情况出现,又会导致表中出现时数据冗余、新增、删除异常等问题出现。

66

因此第四范式的定义就是要消除表中的多值依赖关系。怎么做呢?拿前面的权限表举例。

9

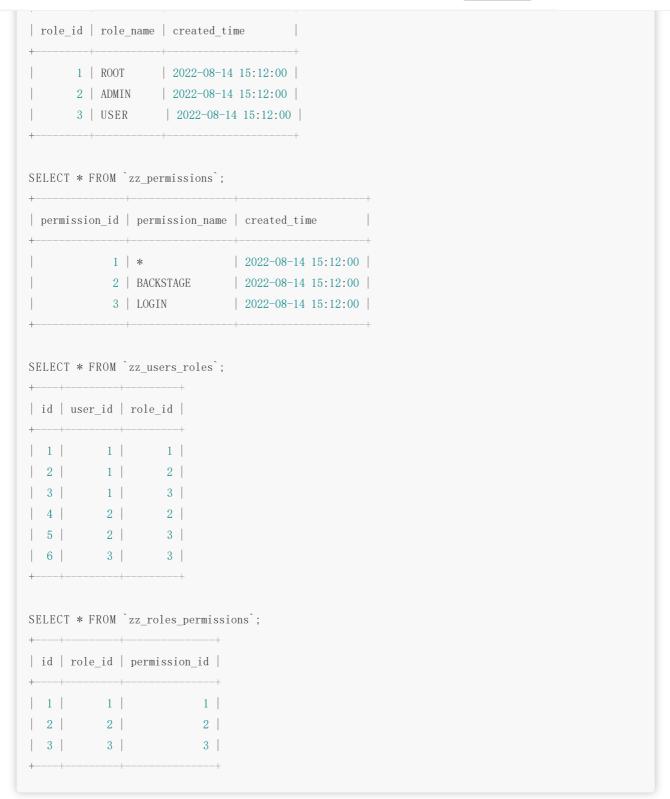












观察上述的五张表,如果有做过权限设计,或用过 Shiro 框架的小伙伴应该会感到额外的亲切,这个正是大名鼎鼎的权限五表,将原本的用户角色权限表,拆分成了用户表、角色表、权限表、用户角色关系表、角色权限关系表。经过这次拆分之后,一方面用户表、角色表、权限表中都不会有数据冗余,第二方面无论是要删除亦或新增一个角色、权限时,都不会影响其他表。

<u>1</u> 53

29









对于前面的教师排课表,就不再拆分啦,大家如若想要锻炼一下掌握程度,可自行将其拆分成符合第四范式的表结构。

∠ 2.3、第五范式 (5NF) /完美范式 (∠



了解了第四范式后,再来看看第五范式,这个范式也被称为完美范式,先来说一下第五范式的定义: 「建立在 4NF 的基础上,进一步消除表中的连接依赖,直到表中的连接依赖都是主键所蕴含的」。等等,连接依赖又是个啥?

连接依赖

4

看不懂对不?说实话我也看着迷糊,大概能确定的是:多值依赖也属于连接依赖的一种,而连接依赖也包含了多值依赖,大家可以参考一下《这个》。

99

第五范式解决的是无损连接问题,但对于第五范式我自个儿也没理解透彻,因此不再讲解第五范式了,防止误导诸位,同时如若有对这块十分了解的大佬,可以留言指点一下。

∠ 2.4、六大范式小结 (∠

经过一系列的阐述后,其实不难发现,越到后面的范式,越难令人理解,同时为了让表满足更高级别的范式,越往后付出代价也越大,而且拆分出的表数量也会越多,所以一般实际开发中,对于库表的设计最高满足 BC 范式即可,再往后就没意义了,因为表数量一多,查询也好,写入也罢,性能会越来越差。

66

同时,由于后面的几种范式在实际项目中应用较少,因此关于这块的资料也会较少,后续的几种范式也仅有一些学术机构在琢磨,所以当我们试图去窥探时,能看到的也是一堆学术词汇。

5

29









范式之间的递进关系图:

心七大范式

• 第一范式: 原子性, 每个字段的值不能再分。

• 第二范式: 唯一性, 表内每行数据必须描述同一业务属性的数据。

• 第三范式: 独立性, 表中每个非主键字段之间不能存在依赖性。

• 巴斯范式: 主键字段独立性, 联合主键字段之间不能存在依赖性。

• 第四范式: 表中字段不能存在多值依赖关系。

• 第五范式: 表中字段的数据之间不能存在连接依赖关系。

• 域键范式: 试图研究出一个库表设计时的终极完美范式。

☲ 三、数据库反范式设计

遵循数据库范式设计的结构优点很明显,它避免了大量的数据冗余,节省了大量存储空间,同时让整体结构更为优雅,能让 SQL 操作更加便捷且减少出错。但随着范式的级别越高,设计出的结构会更加精细化,原本一张表的数据会被分摊到多张表中存储,表的数量随之越来越多。

但随之而来的不仅仅只有好处,也存在一个致命问题,也就是当同时需要这些数据时,只能采用联表查询的形式检索数据,有时候甚至为了一个字段的数据,也需要做一次连表查询才能获得。这其中的开销无疑是花费巨大的,尤其是当连接的表不仅两三张而是很多张时,有可能还会造成索引失效,这种情况带来的资源、时间开销简直是一个噩梦,这会严重地影响整个业务系统的性能。

66

因此,也正是由于上述一些问题,在设计库表结构时,我们不一定要 100% 遵守范式 准则。这种违反数据库范式的设计方法,就被称之为 「反范式设计」。

99

遵循范式设计也好,反范式设计也罢,本身两者之间并没有优劣之分,只要能够对业务更有利,那就可以称之为好的设计方案。范式的目的仅在于让我们设计的结构更优雅合理,有时候在表中多增加一个字段,从数据库的角度来看,数据会存在冗余问题,会让表结构违反范式的定义,但如若能够在实际情况中减少大量的连表查询,这种设计自然也是可取的。

1 53

() 29









优先的原则去设计结构。

99

当然,对于反范式设计也无需再用更多的语言去描述了,因为本质上就是一个概念词,也就是不遵循数据库范式设计的结构,就被称为反范式结构。不过要牢记的一点是:不是所有不遵循数据库范式的结构设计都被称为反范式,反范式设计是指自己知道会破坏范式,但对业务带来好处大于坏处时,刻意设计出破坏范式的结构。



随意设计出的结构,不满足范式要求,同时还无法给业务上带来收益的,这并不被称为反范式设计,反范式设计是一种刻意为之的思想。

99

园 四、库表设计篇总结

在本篇中详细阐述了 DB 库表设计时的一些思想,也就是范式与反范式设计理论,这些理论仅仅只是一套方法论,实际开发过程中,还是需要根据业务来设计出最合适的结构。在文中提及了六种范式,但一般项目中仅需满足到第三范式或 BC 范式即可,因为这个度刚刚好,再往后就会因为过于精细化设计,导致整体性能反而下降。控制到第三范式的级别,一方面数据不会有太多冗余,第二方面也不会对性能影响过大。



同时,如若打破范式的设定能对业务更有利,那也可以违背范式原则去设计。

99

不过虽说这些属于方法论,但认真看下来之后,相信诸位在之后设计库表结构应该会潜意识的遵循一些范式原则,也会尽量的将表结构设计的更为优雅,从而也能让咱们在开发过程中,减少调整库表结构的次数和带来的影响。

66

53

() 29









- 不合理的结构设计会造成的问题:
 - 。 数据冗余, 会浪费一定程度上的存储空间
 - 。 不便于常规 SQL 操作 (例如插入、删除) , 甚至会出现异常
- 合理的结构设计带来的好处:
 - 。 节省空间, SQL 执行时能节省内存空间, 数据存储时能节省磁盘空间
 - 。 数据划分较为合理, DB 性能整体较高, 并且数据也非常完整
 - 。 结构便于维护和进行常规 SQL 操作

分类:

后端

标签:

Java

数据库

MySQL

文章被收录于专栏:



全解MySQL数据库

从MySQL整体架构出发,到SQL优化、MySQL索引、慢查询优化...

关注专栏

相关课程



VIP SpringBoot 源码解读与原理分析

LinkedBear VIV.37

3611购买

¥39.9



VIP Hello,分布式锁

编程界的小...

1506购买

¥29.9

评论

输入评论 (Enter换行, Ctrl + Enter发送)

29

マ收藏









大佬,第四范式那里,课程 + 老师 确定不了 教材啊 竹熊老师 + 语文 对应着两套教材 😭 ,这里是 我的理解有问题么

心点赞 ♀4

竹子爱熊猫 (作者)

1月前

可以确定的,比如给你换个例子,拿一些moba游戏为例,现在出了一个皮肤返场活动, 但是限购,即一个玩家同时只能购买一个英雄的一个皮肤,比如: 英雄A有①、②两款皮肤返场了,英雄B有①、②、③三款皮肤返场了 那这里就存在三个字段, 玩家、英雄、皮肤, 这里我们无法根据英雄、玩家单个字段来 确定玩家到底购买了哪个皮肤,这里必须结合玩家+英雄才能确定,比如玩家1购买了英 雄A的①皮肤。...

展开

心点赞 ♀回复

竹子爱熊猫 (作者)

1月前

目前调整了一下,这样更加合理一点。



心点赞 ♀回复

查看更多回复 >



狗霸天 ❖JY.3 后端开发

1月前

清晰!

心 点赞 ♀ 1

竹子爱熊猫 (作者)

1月前

三克油~,后续文章尽量加快更新。

心点赞 ♀回复



StoneDB ******** 数据库架构师 @ 石原...

1月前

就喜欢看MySQL的文章,整挺好~ 🥸

心点赞 ♀1

~53

29

~7 收藏









Java

CC_LKL *JY.3

1月前

索引这块大概多久写啊,一直对索引体系搞不太明白,搜了一些东西也感觉乱乱的,坐等大佬更新 三(大金牙)

心点赞 ♀4

竹子爱熊猫 (作者)

1月前

这篇文章之后就是索引了哈,索引分为了三篇的,有《索引初识篇》、《索引应用 篇》、《索引原理篇》,分别对应基础、实战以及进阶的,未来十天内会把这三篇更完

△点赞 回复

Java CC LKL 回复 竹子爱熊猫

1月前

好的,期待期待~

"这篇文章之后就是索引了哈,索引分为了三篇的,有《索引初识篇》、《索引应...

△点赞□复

查看更多回复 >

Java

CC_LKL *JY.3

1月前

花了一个多小时,把这篇一点一点看下来了,写的很细致,也很通俗易懂,之前就看那三范式都令 我很头痛,下班的时候就想着看完大佬这篇新文再吃饭,没想到看到现在 😂

161 💬 2

竹子爱熊猫 (作者)

1月前

哈哈哈, 记得按时吃饭, 身体重要~

心点赞 ♀回复

Java CC LKL 回复 竹子爱熊猫

1月前

好勒,感谢竹哥关心 🥯

"哈哈哈,记得按时吃饭,身体重要~"

心 占赞 □ □ □ 恒

29

くく 收藏









大佬新文,来捧捧场 🤒

心点赞 ♀2





竹子爱熊猫 🚧 📆 👣 (作者) Υ掘金签约作者 @ Jav...

1月前

这篇库表设计篇的文章, 更偏向于方法论, 更多的是让大家掌握和熟悉设计库表的原则, 掌握这些 方法论后才能将库表结构设计的更为合理,也能减少开发过程中的出错与重构次数,有不懂的可将 疑惑留在评论区 🥻 ~

162 回8



| 向往ing

1月前

有spring全家桶的专栏吗

心点赞 ♀回复

p 竹子爱熊猫 (作者) 回复 向往ing

1月前

Spring体系比较大,但后续可能会开相应的源码分析专栏,但对于入门专栏应该不会 开, 毕竟网上有很多很多相关的教程。

"有spring全家桶的专栏吗"

心点赞 ♀回复

查看更多回复 >

TU 77 17 4+

29

マ收藏









(十一)MySQL日志篇之undo-log、redo-log、bin-log.....傻傻分不清!

Java中文社群 | 5月前 | 后端 面试 数据库

面试突击52: 什么是三范式? 它有什么用?

傻梦兽 | 8月前 | 前端 · React.js

React 的新范式 - DI, RxJS & Hooks

唐之初 | 10月前 | 前端 · Vue.js

Vue3 setup + TypeScript 开发新范式

宫秋 | 3年前 | 前端 · CSS

可能是最全的前端动效库汇总

谢小飞 | 2年前 | 面试

面试完50个人后我写下这篇总结

神三元 | 2年前 | JavaScript 面试

(建议精读) HTTP灵魂之问,巩固你的 HTTP 知识体系

竹子爱熊猫 | 1月前 | 数据库 MySQL Java

(五)MySQL索引应用篇:建立索引的正确姿势与使用索引的最佳指南!

lecepin | 10月前 | 前端 · 架构

降低前端业务复杂度新视角: 状态机范式

53

() 29









(二)全解MySQL: 一条SQL语句从诞生至结束的多姿多彩历程!

捡田螺的小男孩 | 1年前 | 后端 · Java

美团二面: Redis与MySQL双写一致性如何保证?

⊚ 6.4w 🖒 663 💬 134

合一大师 | 2年前 | React.js

Ant Design从无到有,带你体悟大厂前端开发范式

~ 藕爸~ | 4年前 | 数据库・后端・架构

10亿级订单系统分库分表设计思路!

前端鲨鱼哥 | 1年前 | JavaScript

最全的 Vue 面试题+详解答案

MySQL技术 | 2年前 | MySQL

MySQL库表设计小技巧

◎ 210 心 点赞 ፡ 评论

码农突围 | 2年前 | MySQL

MySQL 常用分库分表方案,都在这里了!

◎ 275 心 点赞 ፡ 评论

Java3y | 4年前 | 数据库 MySQL HTTPS

数据库两大神器【索引和锁】

芋道源码 艿艿 | 5年前 | MySQL · Java · 架构

JDBC PreparedStatement 实现原理【推荐阅读】

53

29









搞懂关系型数据库的三大范式,又是一个能与面试官扯上一会的知识!

◎ 3128 🖒 14 💬 4

民工哥技术之路 | 4年前 | 数据库 · 服务器 · MySQL

MySQL 分库分表方案,总结的非常好!

友情链接:

小贵族 来宝SRV 双环SCEO 主席 default constructor and parameterized constructor in java example split 函数 java oraclesql分页 正则表达式表示空字符串 java架构规范 mysql查找最大 值