6 mysql底层解析——缓存，Innodb_buffer_pool，包括连接、解析、缓存、引擎、存储等

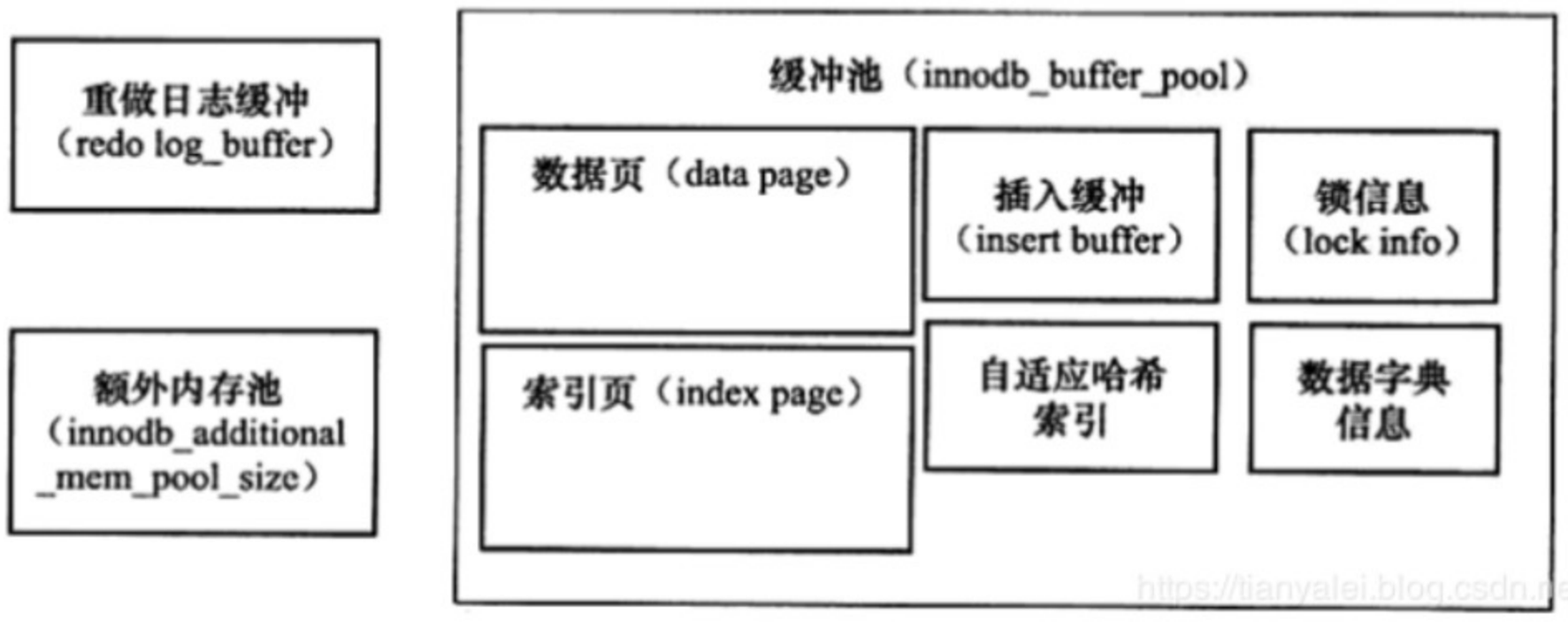
前面几篇主要学习存储，在磁盘上的存储结构，内部格式等。

这一篇就来看内存，对于Innodb来说，也就是最关键的Innodb_buffer_pool。

我们都知道，内存读写和磁盘读写的速度不在一个数量级，在 数据库 中，数据都是最终落到磁盘上的，想要达成快速的读写，必然要依靠缓存技术。

Innodb的这个缓存区就是Innodb_buffer_pool，当读取数据时，就会先从缓存中查看是否数据的页（page）存在，不存在的话才去磁盘上检索，查到后缓存到这个pool里。同理，插入、修改、删除也是先操作缓存里数据，之后再以一定频率更新到磁盘上。控制刷盘的机制，叫做Checkpoint。

Innodb_buffer_pool内部结构



注意，左边那两个不在Innodb_buffer_pool里，是另外一块内存。只不过大部分的内存都属于Innodb_buffer_pool的。

mysql安装后，默认pool的大小是128M，可以通过show variables like 'innodb_buffer_pool%';命令查看。

Variable_name	Value
innodb_buffer_pool_chunk_size	134217728
innodb_buffer_pool_dump_at_shutdown	ON
innodb_buffer_pool_dump_now	OFF
innodb_buffer_pool_dump_pct	25
innodb_buffer_pool_filename	ib_buffer_pool
innodb_buffer_pool_instances	1
innodb_buffer_pool_load_abort	OFF
innodb_buffer_pool_load_at_startup	ON
innodb_buffer_pool_load_now	OFF
innodb_buffer_pool_size	134217728

可以通过show global status like '%innodb_buffer_pool_pages%'；查看已经被占用的和空闲的。共计8000多个page。

Query Favorites　Query History

Variable_name	Value
Innodb_buffer_pool_pages_data	829
Innodb_buffer_pool_pages_dirty	0
Innodb_buffer_pool_pages_flushed	13444
Innodb_buffer_pool_pages_free	7353
Innodb_buffer_pool_pages_misc	9
Innodb_buffer_pool_pages_total	8191

所以如果你的数据很多，而pool很小，那么性能就好不了。

理论上来说，如果你给pool的内存足够大，够装下所有数据，要访问的所有数据都在pool里，那么你的所有请求都是走内存，性能将是最好的，和redis类似。

官方建议pool的空间设置为物理内存的50%-75%。

在mysql5.7.5之后，可以在mysql不重启的情况下动态修改pool的size，如果你设置的pool的size超过了1G的话，应该再修改一下Innodb_buffer_pool_instances=N，将pool分成N个pool实例，将来操作的数据会按照page的hash来映射到不同的pool实例。

这样可以大幅优化多线程情况下，并发读取同一个pool造成的锁的竞争。

缓冲区LRU淘汰算法

当pool的大小不够用了，满了，就会根据LRU算法（最近最少使用）来淘汰老的页面。最频繁使用的页在LRU列表的前端，最少使用的页在LRU列表的尾端。淘汰的话，就首先释放尾端的页。

InnoDB的LRU和普通的不太一样，Innodb的加入了midpoint位置的概念。最新读取到的页，并不是直接放到LRU列表的头部的，而是放到midpoint位置。这个位置大概是LRU列表的5/8处，该参数由innodb_old_blocks_pct控制。

如37是默认值，表示新读取的页插入到LRU尾端37%的位置。在midpoint之后的列表都是old列表，之前的是new列表，可以简单理解为new列表的页都是最活跃的数据。

为什么不直接放头部？因为某些数据扫描操作需要访问的页很多，有时候这些页仅仅在本次查询有效，以后就不怎么用了，并不算是活跃的热点数据。那么真正活跃的还是希望放到头部去，这些新的暂不确定是否真正未来要活跃。所以，这可以理解为预热。还引入了一个参数innodb_old_blocks_time用来表示页读取到mid位置后需要等待多久才会被加入到LRU列表的热端。

还有一个重要的查询命令可以看到这些信息，show engine innodb status;

```
mysql> SHOW ENGINE INNODB STATUS\G;
*************************** 1. row ***************************
  Type: InnoDB
  Name:
Status:
=====================================
120725 22:04:25 INNODB MONITOR OUTPUT
=====================================
Per second averages calculated from the last 24 seconds
......
Buffer pool size   327679
Free buffers       0
Database pages     307717
Old database pages 113570
Modified db pages  24673
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 6448526, not young 0
48.75 youngs/s, 0.00 non-youngs/s
Pages read 5354420, created 239625, written 3486063
55.68 reads/s, 81.74 creates/s, 955.88 writes/s
Buffer pool hit rate 1000 / 1000, young-making rate 0 / 1000 not 0 / 1000
......
```


Database pages表示LRU列表中页的数量，pages made young显示了LRU列表中页移动到前端的次数，Buffer pool hit rate表示缓冲池的命中率，100%表示良好，该值小于95%时，需要考虑是否因为全表扫描引起了LRU列表被污染。里面还有其他的参数，可以自行查阅一下代表什么意思。

Pool的主要空间

上面主要是讲了insert buffer。对应的是增删改时的缓存处理。

从最上面的图能看到，其实更多的、对性能影响更大的是读缓存。毕竟多数数据库是读多写少。

读缓存主要数据是索引页和数据页，这个前面也说过，如果要读取的数据在pool里没有，那就去磁盘读，读到后的新页放到pool的3/8位置，后续根据情况再决定是否放到LRU列表的头部。注意，最小单位是页，哪怕只读一条数据，也会加载整个页进去。如果是顺序读的话，刚好又在同一个页里，譬如读了id=1的，那么再读id=2的时，大概率直接从缓存里读。

插入缓冲insert buffer

从名字就能看出来是干什么的，它是buffer_pool的一部分，用来做insert操作时的缓存的。

我们之前学习过b+ tree，也知道数据的存放格式，那么当新插入数据时，倘若直接就插入到b+ tree里，那么可想会多么缓慢，需要读取、找到要插入的地方，还要做树的扩容、校验、寻址、落盘等等一大堆操作，等你插进去，姑娘都等老了。

在Innodb中，主键是行唯一标识，如果你的插入顺序是按照主键递增进行插入的，那么还好，它不需要磁盘的随机读取，找到了页，就能插，这样速度还是可以的。

然而，如果你的表上有多个别的索引（二级索引），那么当插入时，对于那个二级索引树，就不是顺序的了，它需要根据自己的索引列进行排序，这就需要随机读取了。二级索引越多，那么插入就会越慢，因为要寻找的树更多了。还有，如果你频繁地更新同一条数据，倘若也频繁地读写磁盘，那就不合适了，最好是将多个对同一page的操作，合并起来，统一操作。

所以，Innodb设计了Insert Buffer，对于非聚簇索引的插入、更新操作，不是每次都插入到索引页中，而是先判断该二级索引页是否在缓冲池中，若在，就直接插入，若不在，则先插入一个insert buffer里，再以一定的频率进行真正的插入到二级索引的操作，这时就可以聚合多个操作，一起去插入，就能提高性能。

然而，insert buffer需要同时满足两个条件时，才会被使用：

1 索引是二级索引
2 索引不是unique

注意，索引不能是unique，因为在插入缓冲时，数据库并不去查询索引页来判断插入的记录的唯一性，如果查找了，就又会产生随机读取。

insert buffer的问题是，在写密集的情况下，内存会占有很大，默认最大可以占用1/2的Innodb_buffer_pool的空间。很明显，如果占用过大，就会对其他的操作有影响，譬如能缓存的查询页就变少了。可以通过IBUF_POOL_SIZE_PER_MAX_SIZE来进行控制。

变更缓冲change buffer

新版的Innodb引入了Change buffer，其实就是insert buffer类似的东西，只是把insert、update、delete都进行缓冲。

也就是所有DML操作，都会先进缓冲区，进行逻辑操作，后面才会真正落地。

通过参数Innodb_change_buffering开始查看修改各种buffer的选项。可选值有inserts\deletes\purges\changes\all\none。

```
11 show variables like 'innodb_change_buffering';
```

Query Favorites　Query History

Variable_name	Value
innodb_change_buffering	all

```
8
9  show variables like 'innodb_change_buffer_max_size';
10
11 show variables like 'innodb_change_buffering';
```

Query Favorites | Query History

Variable_name	Value
innodb_change_buffer_max_size	25

默认是所有操作都入buffer，右图的参数是控制内存大小的，25代表最多使用1/4的缓冲池空间。

Insert Buffer原理

insert buffer的 数据结构 是一棵B+ tree，全局就一棵B+树，负责对所有的表的二级索引进行插入缓存。在磁盘上，该tree存放在共享表空间（希望还记得是什么），默认ibdata1中。有时，通过独立表空间的ibd文件试图恢复表中数据时，可能会有CHECK TABLE错误，就是因为该表的二级索引中的数据可能还在insert buffer里，没有刷新到自己的表空间。这时，可以通过repair table来重建表上的所有二级索引。

我们下面来看看这棵B+ tree里是什么样的。

首先，这里存的值将来是要刷到二级索引的，我们至少要知道的信息有：哪个表、表的哪个页面（page）。

所以，insert buffer的b+ tree的内节点（非叶子节点）存放的是查询的search key

space存的是哪个表，offset是所在页的偏移量，可以理解为pageNo。

当发起了一次插入、更新时，首先判断要操作的数据的页（是二级索引的页）是否已经在Innodb_buffer_pool里了，如果在，说明之前可能是查询过该页的数据，既然在缓存了，那就不需要insert buffer了，直接去修改pool里该页的数据即可。

如果不在，那就需要构造search key了，构造好，再加上被插入、修改的数据，插入到insert buffer的叶子节点里去。

何时Merge insert buffer

insert buffer是一棵b+ tree，如果插入、修改的记录的二级索引没在pool里，就需要将记录插到insert buffer。那么什么时候，insert buffer里的数据会被merge到真正的二级索引里去呢?

1 二级索引被读取到pool时

2 insert buffer已无可用空间

3 master thread主线程后台刷入

第一种情况好理解，因为写到insert buffer就是因为该记录的二级索引页不在pool里，现在如果被select到pool里去了，那么自然就会直接merge过去。

第二种情况，那就是没可用空间了，迫不得已，就得去刷磁盘了。

第三种，之前的文章还没提到过，那就是有个master线程每秒或每10秒回进行一次merge insert buffer的操作，不同之处是每次merge的数量不同。

CheckPoint技术（redo log）

上面主要说了insert buffer，它是针对二级索引的插入、修改、删除的缓存，并且是数据页不在pool里才用的。

那如果数据页在pool里，发生了增删改操作后，系统又是何时将数据落地刷入到磁盘呢?

你执行了一条DML语句，pool的页就变成了脏页，因为pool里的比磁盘里的新，两者并不一致。数据库就需要按一定规则将脏页刷入到磁盘。

倘若每次一个页发生变化就刷入磁盘，那开销是非常大的，必须要攒够一定数量或一段时间，再去刷入。还有，pool是内存，倘若还没来得及刷入到磁盘，发生了宕机，那么这些脏页数据就会丢失。

为了解决上面的问题，当前事务数据库系统普通采用了write ahead log策略，即当事务提交时，先写重做日志（redo log），再修改页。当发生故障时，通过redo log来进行数据的恢复。

到这时，基本Innodb的增删改查的流程，基本清晰了。

增删改时，首先顺序写入redo log（顺序写磁盘，类似于kafka），然后修改pool页（pool里没有的，插入insert buffer），之后各种线程，会按照规则从缓存里将数据刷入到磁盘，进行持久化，发生故障了，就从redo log恢复。

checkpoint（检查点）做的事情就是：

1 缓冲池不够用时，将脏页刷新到磁盘

2 redo log不够用时，刷新脏页。

redo log也是磁盘文件，并不能无限增长，而且要循环利用。

Checkpoint所做的事情就是将缓冲池脏页写回磁盘，那么主要就是每次刷多少，每次从哪里去脏页，什么时间去刷的问题了。

目前有两种Checkpoint：

1 数据库关闭时，将所有脏页刷新到磁盘，这是默认的方式。

2 Master Thread操作，这个主线程会每秒、每10秒从脏页列表刷新一定比例的页到磁盘，这是个异步的操作，不会阻塞查询。

3 LRU 列表空闲页不足时，需要刷新一部分来自LRU列表的脏页。

4 redo log文件不可用时，需要强制刷新一部分，为了保证redo log的循环利用。

5 pool空间不足时，脏页太多，需要刷新。

两次写

这是Innodb的一个很独特的功能，是用来保证数据页的可靠性。

我仔细看过后，感觉设计很巧妙，但好像离我们比较远了，所以就不写了，想深入了解的可以自己去查查。

缓存　云数据库 SQL Server　数据库　sql

评论

登录 后参与评论

推荐阅读

编辑精选文章

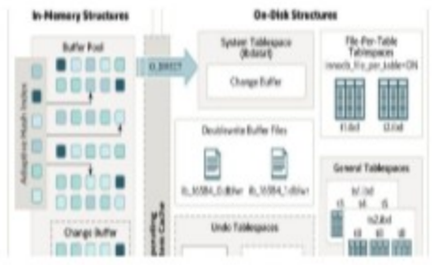
换一批

- 进程，线程，协程 – 你了解… 5029
- 微服务与分布式系统设计看… 4291
- 腾讯文档表格卡顿指标探索… 3218
- 从Hadoop1.0到Hadoop2.0… 3210
- 微服务架构：由浅入深带你… 3213
- 花了1个月学大数据，我想说... 3750

一文搞懂 MySQL InnoDB架构 Buffer Pool、Change Buffer、自适应哈希索引、Log…

索引 buffer mysql pool 架构

书接上回，林渊重生使得 MySQL B+Tree 提前问世二十年，这次亲自操刀 InnoDB 架构引擎的设计，一个支持高并发读写、支持事务行级锁的划时代架构诞生...



码哥字节 · 2025/04/09 7.2K 0

MySQL InnoDB 存储引擎原理浅析

云数据库 SQL Server 缓存 sql

本文主要基于MySQL 5.6以后版本编写，多数知识来着书籍《MySQL技术内幕++InnoDB存储引擎》，今年的多数学习知识只写在笔记里，较为零散，最近稍有时间整理出来，分享进步。

程序员小强 · 2021/05/27 1.6K 0

【精华】洞悉MySQL底层架构：游走在缓冲与磁盘之间

云数据库 SQL Server 数据库 sql https 网络安全

提起MySQL，其实网上已经有一大把教程了，为什么我还要写这篇文章呢，大概是因为网上很多网站都是比较零散，而且描述不够直观，不能对MySQL相关知识有一个系统的学习，导致不能形成知识体系。为此我撰写了这…

cxuan · 2020/06/12 2K 1

技术分享 | Update更新慢、死锁等问题的排查思路分享

缓存 sql server sql 数据库 云数据库 SQL Server

在开始排错之前我们需要知道 Update 在 MySQL 中的生命周期是什么，MySQL 如何执行一个事务的。

老叶茶馆 · 2021/10/12 2.9K 0

详细了解 InnoDB 内存结构及其原理

云数据库 SQL Server 数据库 sql 缓存 编程算法

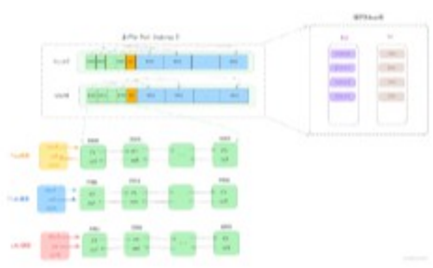
之前写过一篇文章「简单了解InnoDB原理」，现在回过头看，其实里面只是把缓冲池（Buffer Pool），重做日志缓冲（Redo Log Buffer）、插入缓冲（Insert Buffer）和自适应哈希索引（Adaptive Hash Index）等概念简单的...

冬夜先生 · 2021/10/08 654 0

简单聊聊Innodb崩溃恢复那些事

存储 innodb 链表 日志 事务

本文想用简单精炼的语言将Innodb崩溃恢复那些事情好好拾到拾到，本文主要参考以下三本书和我个人一些感想而作:



大忽悠爱学习 · 2023/10/11 773 0

Innodb存储引擎之插入缓冲

存储 云数据库 SQL Server sql 数据库

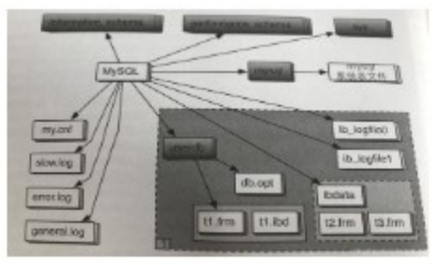
我们知道，innodb存储引擎是基于磁盘存储的，它同时利用缓冲池技术来提高数据库的整体性能，具体的利用方法为：innodb从磁盘中通过16KB数据页的形式，将磁盘中的数据加载到内存当中，通过内存的速度来弥补磁盘...

AsiaYe · 2019/11/06 734 0

3 mysql底层解析——innodb文件系统初步入门，包括连接、解析、缓存、引擎、存...

云数据库 SQL Server 缓存 数据库 sql 存储

上一篇我们学习了server层对于表对象缓存的处理，表对象获取到之后，通过handler才具备了与存储引擎交互的能力。那么存储引擎层又是怎么个流程呢?

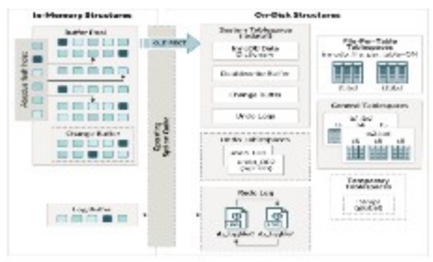


天涯泪小武 · 2019/08/26 960 0

认识InnoDB的Buffer Pool

mysql

对于innoDB存储引擎来说，数据是存储在磁盘上，而执行引擎想要操作数据，必须先将磁盘的数据加载到内存中才能操作。当数据从磁盘中取出后，缓存内存中，下次...



小许code · 2023/06/05 568 0

针对 MySQL/InnoDB 刷盘调优

云数据库 SQL Server 编程算法 数据库 sql https

这篇文章是讲述 InnoDB 刷盘策略系列文章的第三篇。本文主要讲述 性能调优。另外2篇文章参考

用户1278550 · 2022/07/30 2.2K 0

InnoDB Buffer Pool巧配置全解

数据库

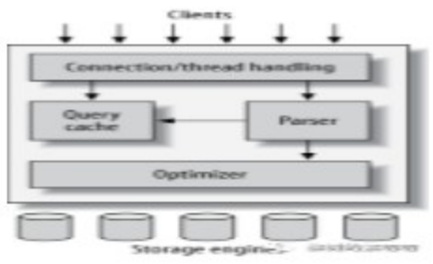
InnoDB 维护了一个缓存数据和索引信息到内存的存储区叫做 Buffer Pool，它会将最近访问的数据缓存到缓冲区。我们通过配置各个 Buffer Pool 的参数，可以显著提高 MySQL 的性能。

阿炳数记 · 2019/02/27 2.4K 0

「干货」MySQL 的 InnoDB 存储引擎是怎么设计的?

sql 数据库 云数据库 SQL Server

上一讲：MySQL 是如何实现 ACID 中的 D 的？ 我用了一个问题，给大家介绍了 MySQL 中的两个成员 binlog 和 redo log。然而，这只是 MySQL 家族里的两个小...



Java3y · 2020/02/19 1.6K 0

答应我，这次要搞懂 Buffer Pool

编程算法 缓存 数据库 sql 云数据库 SQL Server

虽然说 MySQL 的数据是存储在磁盘里的，但是也不能每次都从磁盘里面读取数据，这样性能是极差的。

小林coding · 2022/04/07 765 0

MySQL 存储引擎（2）原

存储 数据库 sql 云数据库 SQL Server 缓存

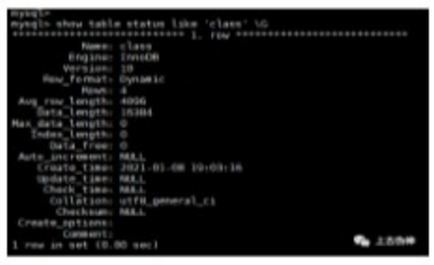
顾名思义，存储引擎就是用于存储我们的数据的。在关系型数据库中我们一般将数据库存放在表中（Table）。

兜兜毛毛 · 2020/04/23 588 0

你真的了解Innodb存储引擎?

云数据库 SQL Server 数据库 sql 存储 缓存

InnoDB的性能和自动崩溃恢复特性，使得它在非事务性存储的需求中也有广泛的应用。

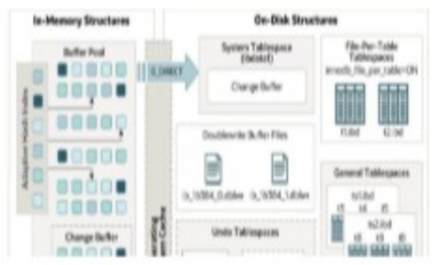


Liusy · 2021/03/03 462 0

MySQL InnoDB架构深度解析

mysql 架构 内存 索引 innodb

MySQL的InnoDB存储引擎是现代Web应用中最常用的数据库存储引擎之一，它以其强大的事务支持、外键约束和并发控制能力而著称。InnoDB的高性能特性很大程度上...

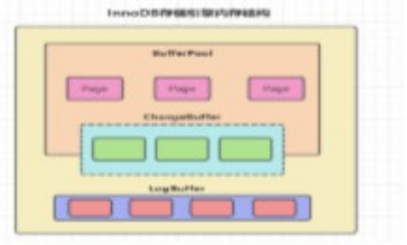


javpower · 2025/05/30 151 0

MySQL的InnoDB存储引擎内存结构

性能优化 innodb mysql

InnoDB内存结构分为三部分，分别是BufferPool缓冲池，ChangeBuffer写缓冲区，LogBuffer日志缓冲区。



Ryan_小鱼 · 2025/03/31 219 0

MySQL InnoDB引擎

数据库 sql mvcc 存储 云数据库 SQL Server

表空间是InnoDB存储引擎逻辑结构的最高层， 如果用户启用了参数innodb_file_per_table(在8.0版本中默认开启)，则每张表都会有一个表空间...

用户9615083 · 2022/12/25 1.6K 0

Innodb Buffer Pool详解

云数据库 SQL Server 文件存储 编程算法 jquery vr 视频解决方案

导读 数据库为了高效读取和存储物理数据，通常都会采用缓存的方式来弥补磁盘IO与CPU运算速度差。InnoDB 作为一个具有高可靠性和高性能的通用存储引擎也不例外...

腾讯数据库技术 · 2023/01/30 1.5K 0

MySQL InnoDB 存储引擎探秘

云数据库 SQL Server mongodb

在MySQL中InnoDB属于存储引擎层，并以插件的形式集成在数据库中。从MySQL5.5.8开始，InnoDB成为其默认的存储引擎。InnoDB存储引擎支持事务、其...

烂猪皮 · 2019/03/12 1.2K 0