

# hw7: MySQL Backup, Recovery & Partition

1. 请你详细描述如何通过全量备份和增量备份来实现系统状态恢复。(2分)
2. 请你根据MySQL缓存的工作原理, 描述预取机制的优点。(1分)
3. 请你按照你的理解, 阐述Partition机制有什么好处? 如果数据文件在一台机器上有足够的存储空间存储, 是否还需要进行Partition?(2分)

## Q1

### 概念介绍

**全量备份 full backup** : MySQL server在给定时间点的**所有**data. 于是通过它可以实现给定时间点的恢复。

实现: MySQL对全量备份有多种方式。

优势: 备份的数据最全面且最完整, 当发生数据丢失灾难时只要用一盘磁带(即灾难发生前一天的备份磁带)就可以恢复全部的数据, 恢复数据的速度快。

缺点: 需要很大的存储空间, 带宽, 完成备份过程相对耗时。

**增量备份 incremental backup** : 在一个时间段(两个时间点之间)增量式的备份所有的对数据的改变。

实现: 通过 server的binary log 来记录数据改变, 也叫做point-in-time recovery 因为通过全量+增量可以实现任意时间点的恢复。

优势: 备份速度快, 没有重复的备份数据, 节省了磁带空间, 缩短了备份时间。

缺点: 恢复需要把多个备份集合的数据拼凑在一起, 如果一个备份集crash, 完全恢复的可能性较小。

**系统恢复通过结合 全量备份和增量备份来实现:**

1. 比如现在11月3日12:05 数据库崩溃了需要恢复到之前的状态。
2. 查找所有的全量备份, 看到了最新的11月1日 00:00做了全量备份, 就先恢复到这个状态
3. 找到11月1日00:00-11月2日:00 和11月2日00:00-11月3日00:00 的已经写完成了bin-log文件(这两个文件不会修改了), 按照顺序恢复
4. 再找到现在正在写的(11月3日00:00-12:05) bin-log 文件进行恢复。

这样通过全量备份和增量备份就可以恢复到任意的时间点了。然而需要注意的是，无论是全量还是增量备份，都需要存到非这个数据库的机器/硬盘上，否则可能一起crash, 这样就没有意义了。

## Q2

好处：总结自讲课和板书

目的：通过预读操作控制InnoDB的预取量。

背景：磁盘数据被optimize之后，变成连续存放，内部碎片减少。当系统有未使用的I/O容量时，更多的预读可以提高查询的性能。比如磁盘转一圈就可以把整个表的数据或者使用的数据的周围数据都读出来。所以预读可以提高性能。但是过多的预读可能会导致高负载系统的性能周期性下降。

对于线性预读，基于数据locality，因为是在一个表里做操作，临近数据被取用或者整个表的数据被频繁使用是很可能的。

对于随机预读，上课提到的背景是：但是如果是交易性的操作，只要读一个订单，就应该使用随机处理（因为不需要周围其他的order）。有一张order表，关联了orderItem, orderItem又关联了book，需要一定的预测机制来得到对应的orderItem和book。在这里，预测算法就显得非常重要了，预测的越准，性能越好。

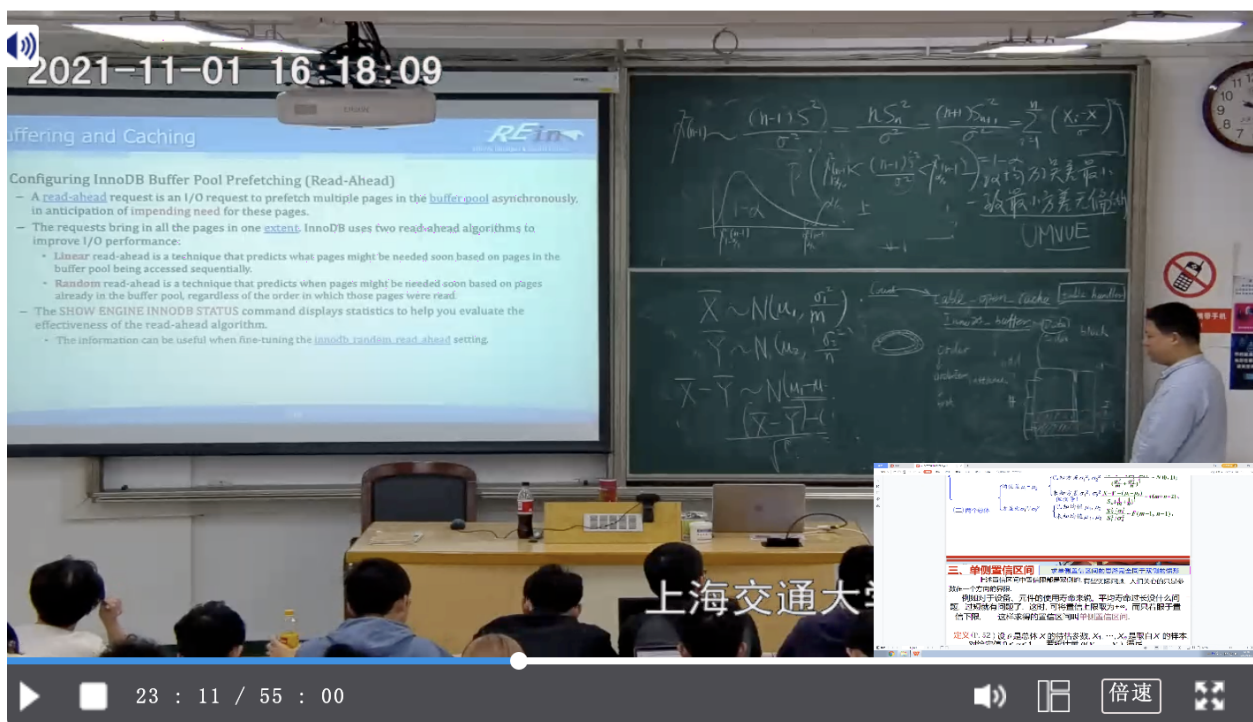
技术：总结自mysql官方文档和PPT

预读请求是指预取缓冲池中的多个页面的异步请求，预计这些页面很快就会被使用。请求在一个区段中引入所有页面。InnoDB使用两种预读算法来提高I/O性能：

**线性预读**是一种基于按顺序访问的缓冲池中的页面来预测可能很快需要哪些页面的技术。通过配置参数innodb\_read\_ahead\_threshold，可以通过调整触发异步读请求所需的顺序页访问次数来控制InnoDB执行预读操作的时间。在此之前，InnoDB只会在读取当前extent的最后一页时，计算是否对整个下一个extent发出异步预取请求。

**随机预读**是一种技术，它可以根据缓冲池中已经存在的页面预测何时可能需要页面，而不管这些页面的读取顺序如何。如果在缓冲池中发现同一个区段的13个连续页面，InnoDB会异步发出一个请求来预取该区段的剩余页面。

- Configuring InnoDB Buffer Pool Prefetching (Read-Ahead)
  - A **read-ahead** request is an I/O request to prefetch multiple pages in the **buffer pool** **asynchronously**, in anticipation of **impending need** for these pages.
  - The requests bring in all the pages in one **extent**. InnoDB uses two read-ahead algorithms to improve I/O performance:
    - Linear** read-ahead is a technique that predicts what pages might be needed soon based on pages in the buffer pool being accessed sequentially.
    - Random** read-ahead is a technique that predicts when pages might be needed soon based on pages already in the buffer pool, regardless of the order in which those pages were read.
  - The **SHOW ENGINE INNODB STATUS** command displays statistics to help you evaluate the effectiveness of the read-ahead algorithm.
    - The information can be useful when fine-tuning the **innodb\_random\_read\_ahead** setting.



## Q3

### 分区的好处：

(1) 可伸缩性：

可能表太大了，单个文件或者硬盘放不下。文件一次性读进来占用很多时间和内存。


将数据分区放在不同磁盘，可以解决单磁盘容量瓶颈问题，存储更多的数据，也能解决单磁盘的IO瓶颈问题。

(2) 提升数据库的性能：



mysql 预读\_MySQL InnoDB缓冲池预读\_weixin\_27038261的博客-CSDN博客

InnoDB在I/O的优化上有个比较重要的特性为预读，预读请求是一个i/o请求，它会异步地在缓冲池中预先回迁多个页面，预计很快就会需要这些页面，这些请求在一个范围内引入所有页面。InnoDB以64个page为一个extent，那么InnoDB的预读是以page为单位

 [https://blog.csdn.net/weixin\\_27038261/article/details/113190221](https://blog.csdn.net/weixin_27038261/article/details/113190221)

原创

#### 关于MySQL buffer pool的预读机制

数据库请求数据的时候，会将读请求交给文件系统，放入请求队列中；相关进程从请求队列中将读请求取出，根据需求到相关数据区(内存、磁盘)读取数据；取出的数据，放入响应队列中，最后数据库就会从响应队列中将数据取

 <https://www.cnblogs.com/geaozhang/p/7397699.html>

#### App开发者高效成长

 增长变现闭环

 收入提升 **28%**

[立即注册](#)



#### MySQL :: MySQL 5.7 Reference Manual :: 14.8.3.4 Configuring InnoDB Buffer Pool Prefetching (Read-Ahead)

A read-ahead request is an I/O request to prefetch multiple pages in the buffer pool asynchronously, in anticipation that these pages are needed soon. The requests bring in all the pages in one extent. InnoDB uses two read-ahead algorithms to improve I/O performance: Linear read-ahead is a technique that predicts what pages might be needed soon based on pages

 [https://dev.mysql.com/doc/refman/5.7/en/innodb-performance-read\\_ahead.html](https://dev.mysql.com/doc/refman/5.7/en/innodb-performance-read_ahead.html)

官方文档对于**Configuring InnoDB Buffer Pool Prefetching (Read-Ahead)**的介绍如下

A read-ahead request is an I/O request to prefetch multiple pages in the buffer pool asynchronously, in anticipation that these pages are needed soon. The requests bring in all the pages in one extent. **InnoDB** uses two read-ahead algorithms to improve I/O performance:

**Linear** read-ahead is a technique that predicts what pages might be needed soon based on pages in the buffer pool being accessed sequentially. You control when **InnoDB** performs a read-ahead operation by adjusting the number of sequential page accesses required to trigger an asynchronous read request, using the configuration parameter `innodb_read_ahead_threshold`. Before this parameter was added, **InnoDB** would only calculate whether to issue an asynchronous prefetch request for the entire next extent when it read the last page of the current extent.

The configuration parameter `innodb_read_ahead_threshold` controls how sensitive **InnoDB** is in detecting patterns of sequential page access. If the number of pages read sequentially from an extent is greater than or equal to `innodb_read_ahead_threshold`, **InnoDB** initiates an asynchronous read-ahead operation of the entire following extent. `innodb_read_ahead_threshold` can be set to any value from 0-64. The default value is 56. The higher the value, the more strict the access pattern check. For example, if you set the value to 48, **InnoDB** triggers a linear read-ahead request only when 48 pages in the current extent have been accessed sequentially. If the value is 8, **InnoDB** triggers an asynchronous read-ahead even if as few as 8 pages in the extent are accessed sequentially. You can set the value of this parameter in the MySQL configuration file, or

change it dynamically with the `SET GLOBAL` statement, which requires privileges sufficient to set global system variables. See [Section 5.1.8.1, “System Variable Privileges”](#).

**Random** read-ahead is a technique that predicts when pages might be needed soon based on pages already in the buffer pool, regardless of the order in which those pages were read. If 13 consecutive pages from the same extent are found in the buffer pool, **InnoDB** asynchronously issues a request to prefetch the remaining pages of the extent. To enable this feature, set the configuration variable `innodb_random_read_ahead` to `ON`.

The `SHOW ENGINE INNODB STATUS` command displays statistics to help you evaluate the effectiveness of the read-ahead algorithm. Statistics include counter information for the following global status variables:

- `Innodb buffer pool read ahead`
- `Innodb buffer pool read ahead evicted`
- `Innodb buffer pool read ahead rnd`

This information can be useful when fine-tuning the `innodb_random_read_ahead` setting.

For more information about I/O performance, see [Section 8.5.8, “Optimizing InnoDB Disk I/O”](#) and [Section 8.12.2, “Optimizing Disk I/O”](#).

#### MySQL :: MySQL 5.7 Reference Manual :: 8.5.9 Optimizing InnoDB Configuration Variables

Different settings work best for servers with light, predictable loads, versus servers that are running near full capacity all the time, or that experience spikes of high activity. Because the InnoDB storage engine performs many of its optimizations automatically, many performance-tuning tasks involve monitoring to ensure that the database is performing well, and

 <https://dev.mysql.com/doc/refman/5.7/en/optimizing-innodb-configuration-variables.html>

- Controlling the amount of **prefetching** that InnoDB does with its read-ahead operations. When the system has unused I/O capacity, more read-ahead can improve the performance of queries. Too much read-ahead can cause periodic drops in performance on a heavily loaded system. See [Section 14.8.3.4, “Configuring InnoDB Buffer Pool Prefetching \(Read-Ahead\)”](#).

👉 提到