# 1.0

打开你手头的数据库，去了解这个数据库的版本演变、 SQL引擎、事务处理级别划分、以及体系结构。

mysql 版本历史演变

* **1995年**，MySQL 1.0发布，仅供内部使用。
* **1996年**，MySQL 3.11.1发布，直接跳过了MySQL 2.x版本。
* 1999年，MySQL AB公司成立。同年，发布MySQL 3.23，该版本集成了Berkeley DB存储引擎。该引擎由Sleepycat公司开发，支持事务。在集成该引擎的过程中，对源码进行了改造，为后续可插拔式存储引擎架构奠定了基础。
* **2000年**，ISAM升级为MyISAM存储引擎。同年，MySQL基于GPL协议开放源码。
* 2002年，MySQL 4.0发布，集成了后来大名鼎鼎的InnoDB存储引擎。该引擎由Innobase公司开发，支持事务，支持行级锁，适用于OLTP等高并发场景。
* 2005年，MySQL 5.0发布，开始支持游标，存储过程，触发器，视图，XA事务等特性。同年，Oracle收购Innobase公司。
* 2008， Sun以10亿美金收购MySQL AB。同年，发布MySQL 5.1，其开始支持定时器（Event scheduler），分区，基于行的复制等特性。
* 2009，Oracle以74亿美金收购Sun公司。
* 2010，MySQL 5.5发布，其包括如下重要特性及更新。 InnoDB代替MyISAM成为MySQL默认的存储引擎。 多核扩展，能更充分地使用多核CPU。 InnoDB的性能提升，包括支持索引的快速创建，表压缩，I/O子系统的性能提升，PURGE操作从主线程中剥离出来，Buffer Pool可拆分为多个Instances。 半同步复制。 引入utf8mb4字符集，可用来存储emoji表情。 引入metadata locks（元数据锁）。 分区表的增强，新增两个分区类型：RANGE COLUMNS和LIST COLUMNS。 MySQL企业版引入线程池。 可配置IO读写线程的数量（innodb*read*io*threads，innodb*write*io*threads）。在此之前，其数量为1，且不可配置。 引入innodb*io*capacity选项，用于控制脏页刷新的数量。
* 2013，MySQL 5.6发布，其包括如下重要特性及更新。 GTID复制。 无损复制。 延迟复制。 基于库级别的并行复制。 mysqlbinlog可远程备份binlog。 对TIME, DATETIME和TIMESTAMP进行了重构，可支持小数秒。DATETIME的空间需求也从之前的8个字节减少到5个字节。 Online DDL。ALTER操作不再阻塞DML。 可传输表空间（transportable tablespaces）。 统计信息的持久化。避免主从之间或数据库重启后，同一个SQL的执行计划有差异。 全文索引。 InnoDB Memcached plugin。 EXPLAIN可用来查看DELETE，INSERT，REPLACE，UPDATE等DML操作的执行计划，在此之前，只支持SELECT操作。 分区表的增强，包括最大可用分区数增加至8192，支持分区和非分区表之间的数据交换，操作时显式指定分区。 Redo Log总大小的限制从之前的4G扩展至512G。 Undo Log可保存在独立表空间中，因其是随机IO，更适合放到SSD中。但仍然不支持空间的自动回收。 可dump和load Buffer pool的状态，避免数据库重启后需要较长的预热时间。 InnoDB内部的性能提升，包括拆分kernel mutex，引入独立的刷新线程，可设置多个purge线程。 优化器性能提升，引入了ICP，MRR，BKA等特性，针对子查询进行了优化。 可以说，MySQL 5.6是MySQL历史上一个里程碑式的版本，这也是目前生产上应用得最广泛的版本。
* 2015，MySQL 5.7发布，其包括如下重要特性及更新。 组复制 InnoDB Cluster 多源复制 增强半同步（AFTER*SYNC） 基于WRITESET的并行复制。 在线开启GTID复制。 在线设置复制过滤规则。 在线修改Buffer pool的大小。 在同一长度编码字节内，修改VARCHAR的大小只需修改表的元数据，无需创建临时表。 可设置NUMA架构的内存分配策略（innodb*numa\_interleave）。 透明页压缩（Transparent Page Compression）。 UNDO表空间的自动回收。 查询优化器的重构和增强。 可查看当前正在执行的SQL的执行计

Mysql引擎

InnoDB主要特性

* 为MySQL提供了具有提交、回滚和崩溃恢复能力的事物安全（ACID兼容）存储引擎。InnoDB锁定在行级并且也在 SELECT语句中提供一个类似Oracle的非锁定读。这些功能增加了多用户部署和性能。在SQL查询中，可以自由地将InnoDB类型的表和其他MySQL的表类型混合起来，甚至在同一个查询中也可以混合
* InnoDB存储引擎为在主内存中缓存数据和索引而维持它自己的缓冲池。InnoDB将它的表和索引在一个逻辑表空间中，表空间可以包含数个文件（或原始磁盘文件）。这与MyISAM表不同，比如在MyISAM表中每个表被存放在分离的文件中。InnoDB表可以是任何尺寸，即使在文件尺寸被限制为2GB的操作系统上
* InnoDB支持外键完整性约束，存储表中的数据时，每张表的存储都按主键顺序存放，如果没有显示在表定义时指定主键，InnoDB会为每一行生成一个6字节的ROWID，并以此作为主键

**MyISAM主要特性：**

* 被大文件系统和操作系统支持
* 当把删除和更新及插入操作混合使用的时候，动态尺寸的行产生更少碎片。这要通过合并相邻被删除的块，若下一个块被删除，就扩展到下一块自动完成
* 每个MyISAM表\*\*\*索引数是64，这可以通过重新编译来改变。每个索引\*\*\*的列数是16
* \*\*\*的键长度是1000字节，这也可以通过编译来改变，对于键长度超过250字节的情况，一个超过1024字节的键将被用上
* BLOB和TEXT列可以被索引
* NULL被允许在索引的列中，这个值占每个键的0~1个字节
* 所有数字键值以高字节优先被存储以允许一个更高的索引压缩
* 每个MyISAM类型的表都有一个AUTOINCREMENT的内部列，当INSERT和UPDATE操作的时候该列被更新，同时AUTOINCREMENT列将被刷新。所以说，MyISAM类型表的AUTOINCREMENT列更新比InnoDB类型的AUTOINCREMENT更快
* 可以把数据文件和索引文件放在不同目录
* 每个字符列可以有不同的字符集
* 有VARCHAR的表可以固定或动态记录长度
* VARCHAR和CHAR列可以多达64KB

**MEMORY存储引擎**

MEMORY存储引擎将表中的数据存储到内存中，未查询和引用其他表数据提供快速访问。

MEMORY主要特性：

* MEMORY表的每个表可以有多达32个索引，每个索引16列，以及500字节的\*\*\*键长度
* MEMORY存储引擎执行HASH和BTREE缩影
* 可以在一个MEMORY表中有非唯一键值
* MEMORY表使用一个固定的记录长度格式
* MEMORY不支持BLOB或TEXT列
* MEMORY支持AUTO\_INCREMENT列和对可包含NULL值的列的索引
* MEMORY表在所由客户端之间共享（就像其他任何非TEMPORARY表）
* MEMORY表内存被存储在内存中，内存是MEMORY表和服务器在查询处理时的空闲中，创建的内部表共享
* 当不再需要MEMORY表的内容时，要释放被MEMORY表使用的内存，应该执行 DELETE FROM或 TRUNCATE TABLE，或者删除整个表（使用DROP TABLE）

# 2.1

Q: 对大多数码农而言，数据库锁机制好像都是自动和透明实现的，那么深入了解每个数据库的锁机制实现细节，对码农编码有什么影响嘛？

可以帮助程序员提前避免死锁等问题，避免一些不必要的bug

Q: 根据这节课Oracle的锁机制特征的分析，你尝试去了解一下MySQL、SQLServer这些其它数据库锁机制实现的特征

MyISAM 表锁

MyISAM表级锁模式：

* 表共享读锁 （Table Read Lock）：不会阻塞其他用户对同一表的读请求，但会阻塞对同一表的写请求；
* 表独占写锁 （Table Write Lock）：会阻塞其他用户对同一表的读和写操作；

MyISAM 表的读操作与写操作之间，以及写操作之间是串行的。当一个线程获得对一个表的写锁后， 只有持有锁的线程可以对表进行更新操作。 其他线程的读、 写操作都会等待，直到锁被释放为止。

InnoDB锁模式：

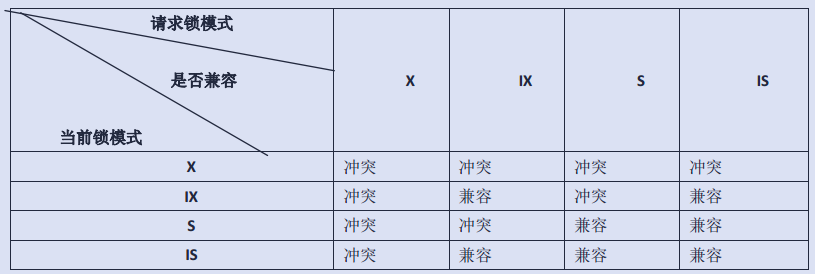
InnoDB 实现了以下两种类型的**行锁**：

* 共享锁（S）：允许一个事务去读一行，阻止其他事务获得相同数据集的排他锁。
* 排他锁（X）：允许获得排他锁的事务更新数据，阻止其他事务取得相同数据集的共享读锁和排他写锁。

为了允许行锁和表锁共存，实现多粒度锁机制，InnoDB 还有两种内部使用的意向锁（Intention Locks），这两种意向锁都是**表锁**：

* 意向共享锁（IS）：事务打算给数据行加行共享锁，事务在给一个数据行加共享锁前必须先取得该表的 IS 锁。
* 意向排他锁（IX）：事务打算给数据行加行排他锁，事务在给一个数据行加排他锁前必须先取得该表的 IX 锁。

**锁模式的兼容情况：**



（如果一个事务请求的锁模式与当前的锁兼容， InnoDB 就将请求的锁授予该事务； 反之， 如果两者不兼容，该事务就要等待锁释放。）

# 2.2

Q: 你对你常用的关系数据库关系系统中，去寻找一些针对优化的工具，去尝试使用一些性能的分析和监控工具（查看数据库官方Reference，首先使用官方的命令和工具）

Pt-qurey-digest

pt-query-digest主要用来分析mysql的慢日志，与mysqldumpshow工具相比，pt-querydigest 工具的分析结果更具体，更完善。pt-querydigest是PT工具集的子集。

1、直接分析慢查询文件:

pt-query-digest /var/lib/mysql/slowtest-slow.log > slow\_report.log

2、分析最近12小时内的查询：

pt-query-digest --since=12h /var/lib/mysql/slowtest-slow.log > slow\_report2.log

3、分析指定时间范围内的查询：

pt-query-digest /var/lib/mysql/slowtest-slow.log --since '2017-01-07 09:30:00' --until '2017-01-07 10:00:00'> > slow\_report3.log

4、分析指含有select语句的慢查询

pt-query-digest --filter '$event->{fingerprint} =~ m/^select/i' /var/lib/mysql/slowtest-slow.log> slow\_report4.log

5、针对某个用户的慢查询。

pt-query-digest --filter '($event->{user} || "") =~ m/^root/i' /var/lib/mysql/slowtest-slow.log> slow\_report5.log

6、查询所有所有的全表扫描或full join的慢查询

pt-query-digest --filter '(($event->{Full\_scan} || "") eq "yes") ||(($event->{Full\_join} || "") eq "yes")' /var/lib/mysql/slowtest-slow.log> slow\_report6.log

Innotop

innotop是一个MySQL和InnoDB事务/状态监视器，它显示查询、InnoDB事务、锁等待、死锁、外键错误，打开表，复制状态，缓冲区信息，行操作、日志、I/O操作、加载图等等。你可以使用innotop同时监控多个服务器。innotp可以综合了解你的 MySQL。

基本使用

innotop --host 192.168.1.181 --user admin --password 123456 --port 3306 --delay 1 -m Q

Q: 关于把数据库当成黑盒使用的错误，其实也会在你学习软件开发中遇到类似的问题，比如，对操作系统的黑盒化，比如对某些开发框架的黑盒化等等，请你思考一下，你的学习过程中，还能找到类似的例子嘛？

如，对java虚拟机模型了解不够深入，有时候创建了许多的大对象，导致jvm频繁发生GC（甚至出现OOM异常），但是回收效果不佳，合理的做法是可以使用单例模式，大对象只加载一次，后面调用这一个对象即可。

# 3.1

* 索引在很多情况下都能提高查询效率，但是相应的也会付出空间占用大的代价。而且错用或者滥用索引，甚至可能造成降低查询效率的结果，且不易被检查出来。
* MyISAM使用前缀压缩减少索引的大小，从而让更多的索引可以放进内存中，这在某些情况下能极大的提高性能。
* InnoDB表数据文件本身就是一个索引结构，树的叶节点data域保存了完整的数据记录，这种索引叫做聚集索引。聚集索引这种实现方式使得按主键的搜索十分高效，但是辅助索引搜索需要检索两遍索引：首先检索辅助索引获得主键，然后用主键到主索引中检索获得记录。

# 3.2

mysql

* B-Tree索引

适合：全键值，键值范围或者键前缀查找。

* Hash索引
* 适合：等值查询。
* 使用限制：无法用于排序、不支持部分索引列匹配查找
* 空间数据索引
* 可以用于地理数据的存储，这类数据无需前缀查询，空间索引会从所有维度来索引数据。

4.1

用的是MySql，没有这个问题。