



高并发高性能 数据库设计-核心篇



• 讲师 : KimmKing •

目录 | Contents

1. 谈谈数据存储
2. 关系数据库与 SQL 语言
3. MySQL 数据库
4. 深入数据库原理
5. MySQL 配置优化经验
6. 数据库设计优化经验
7. MySQL 事务原理
8. MySQL 对分布式事务 XA 的支持



一、谈谈数据存储

计算机组成：CPU、内存、磁盘

应用系统：Network、Application System、Database

从账本，到数据文件，到数据管理系统 ==> 数据库



一、谈谈数据存储

业务规模发展，IT 信息化，导致数据越来越大

技术进步，磁盘容量越来越大、性能越来越好

软盘 -> Sata -> SSD -> Persistent Memory



二、关系数据库与 SQL 语言

- 1970年 Codd 提出关系模型，以关系代数理论为数学基础
《A Relational Model of Data for Large Shared Data Banks》

mathematical sense. Given sets S_1, S_2, \dots, S_n (not necessarily distinct), R is a relation on these n sets if it is a set of n -tuples each of which has its first element from S_1 , its second element from S_2 , and so on.¹ We shall refer to S_j as the j th *domain* of R . As defined above, R is said to have *degree n*. Relations of degree 1 are often called *unary*, degree 2 *binary*, degree 3 *ternary*, and degree n *n-ary*.

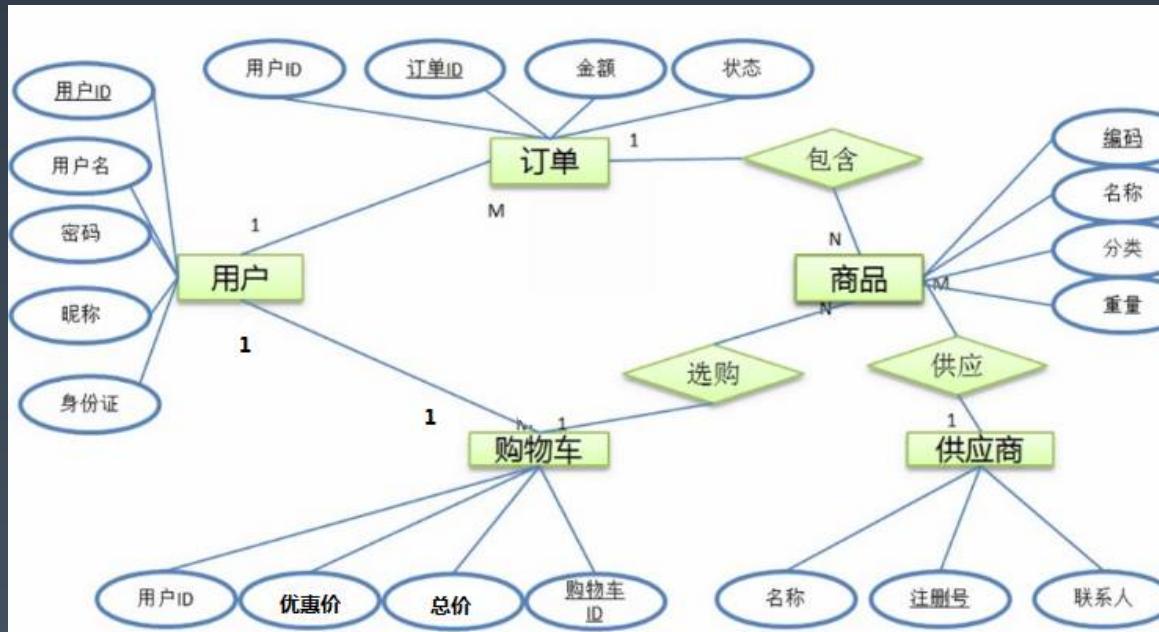
¹ More concisely, R is a subset of the Cartesian product $S_1 \times S_2 \times \dots \times S_n$.

ray which represents an n -ary relation R has the following properties:

- (1) Each row represents an n -tuple of R .
- (2) The ordering of rows is immaterial.
- (3) All rows are distinct.
- (4) The ordering of columns is significant—it corresponds to the ordering S_1, S_2, \dots, S_n of the domains on which R is defined (see, however, remarks below on domain-ordered and domain-unordered relations).
- (5) The significance of each column is partially conveyed by labeling it with the name of the corresponding domain.

二、关系数据库与 SQL 语言

- E-R 图，通过实体-关系，设计数据库结构



二、关系数据库与 SQL 语言

数据库设计范式

第一范式 (1NF) : 关系 R 属于第一范式 , 当且仅当 R 中的每一个属性A的值域只包含原子项

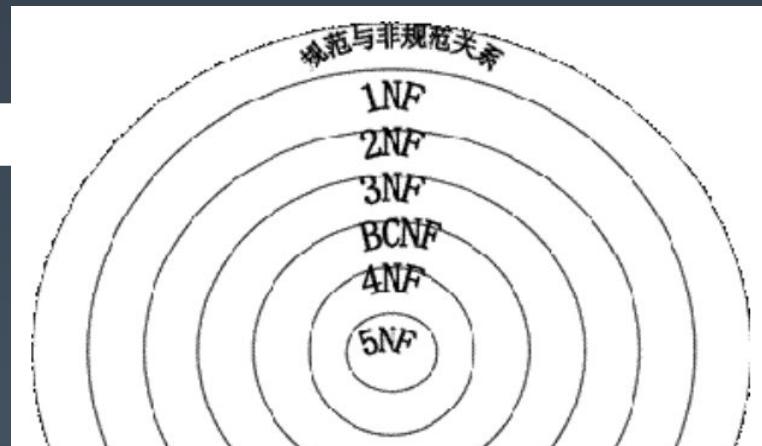
第二范式 (2NF) : 在满足 1NF 的基础上 , 消除非主属性对码的部分函数依赖

第三范式 (3NF) : 在满足 2NF 的基础上 , 消除非主属性对码的传递函数依赖

BC 范式 (BCNF) : 在满足 3NF 的基础上 , 消除主属性对码的部分和传递函数依赖

第四范式 (4NF) : 消除非平凡的多值依赖

第五范式 (5NF) : 消除一些不合适的连接依赖



二、关系数据库与 SQL 语言

数据库设计范式

1NF：消除重复数据，即每一列都是不可再分的基本数据项；

每个列都是原子的。

学号	姓名	系名	系主任	课名	分数
1022211101	李小明	经济系	王强	高等数学	95
1022211101	李小明	经济系	王强	大学英语	87
1022211101	李小明	经济系	王强	普通化学	76
1022211102	张莉莉	经济系	王强	高等数学	72
1022211102	张莉莉	经济系	王强	大学英语	98
1022211102	张莉莉	经济系	王强	计算机基础	88
1022511101	高芳芳	法律系	刘玲	高等数学	82
1022511101	高芳芳	法律系	刘玲	法学基础	82

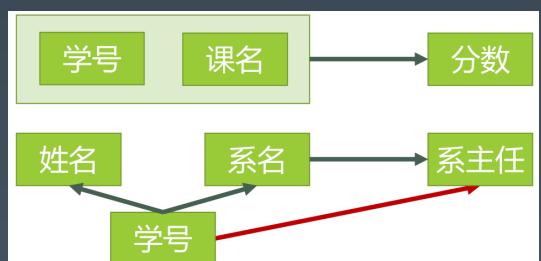
二、关系数据库与 SQL 语言

数据库设计范式

2NF：消除部分依赖，没有列只与主键的部分相关。
即每一行都被主键唯一标识；每个列都有主键。



学号	课名	分数
1022211101	高等数学	95
1022211101	大学英语	87
1022211101	普通化学	76
1022211102	高等数学	72
1022211102	大学英语	98
1022211102	计算机基础	88
1022511101	高等数学	82
1022511101	法学基础	82

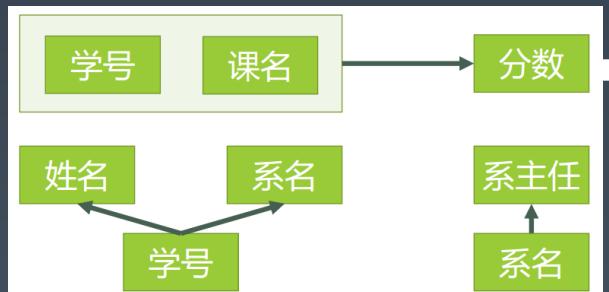


学号	姓名	系名	系主任
1022211101	李小明	经济系	王强
1022211102	张莉莉	经济系	王强
1022511101	高芳芳	法律系	刘玲

二、关系数据库与 SQL 语言

数据库设计范式

3NF：消除传递依赖，消除表中列不依赖主键，而是依赖表中非主键列的情况，即没有列是与主键不相关的。
从表只引用主表的主键，即每列都和主键相关。



学号	课名	分数
1022211101	高等数学	95
1022211101	大学英语	87
1022211101	普通化学	76
1022211102	高等数学	72
1022211102	大学英语	98
1022211102	计算机基础	88
1022511101	高等数学	82
1022511101	法学基础	82

学号	姓名	系名
1022211101	李小明	经济系
1022211102	张莉莉	经济系
1022511101	高芳芳	法律系

系名	系主任
经济系	王强
经济系	王强
法律系	刘玲

二、关系数据库与 SQL 语言

数据库设计范式

BCNF : Boyce-Codd Normal Form (巴斯-科德范式)

3NF 的基础上消除主属性对于码的部分与传递函数依赖。

仓库名	管理员	物品名	数量
上海仓	张三	iPhone 5s	30
上海仓	张三	iPad Air	40
北京仓	李四	iPhone 5s	50
北京仓	李四	iPad Mini	60

仓库 (仓库名 , 管理员)

库存 (仓库名 , 物品名 , 数量)

二、关系数据库与 SQL 语言

常见关系数据库

开源 : MySQL、PostgreSQL

商业 : Oracle , DB2 , SQL Server

内存数据库 : Redis ? , VoltDB

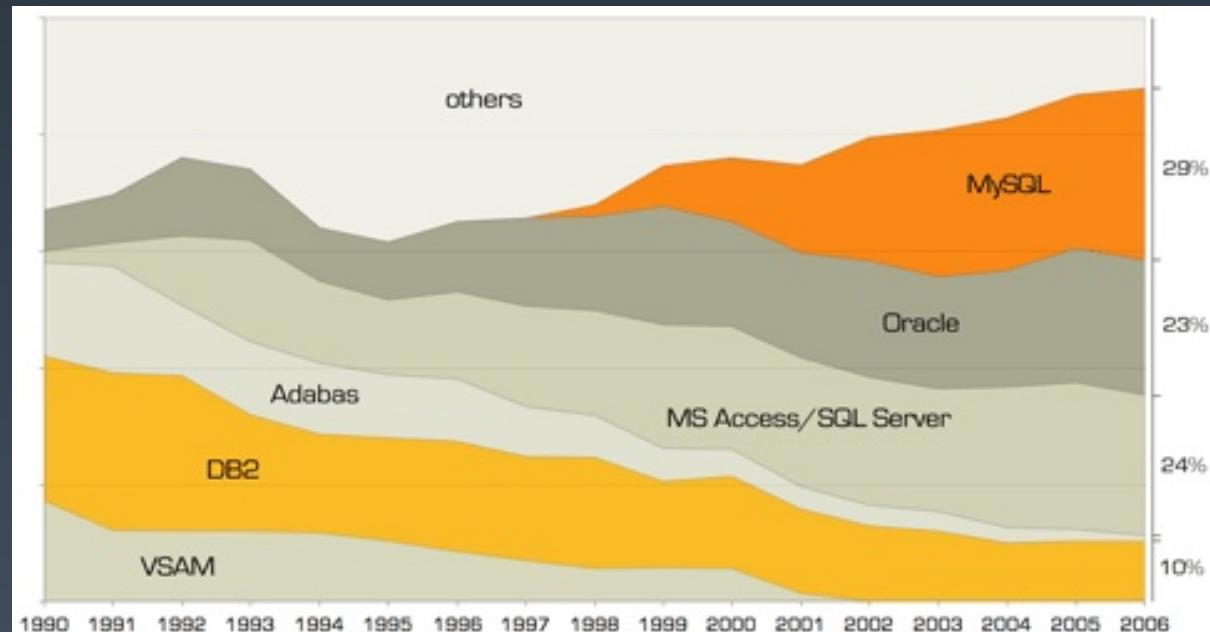
图数据库 : Neo4j , Nebula

时序数据库 : InfluxDB、openTSDB

其他关系数据库 : Access、Sqlite、H2、Derby、Sybase、Infomix 等

NoSQL 数据库 : MongoDB、Hbase、Cassandra、CouchDB

NewSQL/ 分布式数据库 : TiDB、CockroachDB、NuoDB、OpenGauss、OB、TDSQL



二、关系数据库与 SQL 语言

SQL 语言1974年由 Boyce 和 Chamberlin 提出，并首先在 IBM 公司研制的关系数据库系统 SystemR 上实现。

1979年 ORACLE 公司首先提供商用的 SQL，IBM 公司在 DB2 和 SQL/DS 数据库系统中也实现了 SQL。

1986年10月，美国 ANSI 采用 SQL 作为关系数据库管理系统的标准语言（ANSI X3.135-1986），后为国际标准化组织（ISO）采纳为国际标准。

1989年，美国 ANSI 采纳在 ANSI X3.135-1989报告中定义的关系数据库管理系统的 SQL 标准语言，称为 ANSI SQL 89，该标准替代 ANSI X3.135-1986 版本。

思考：SQL 语言是不是操作数据库必须的？

二、关系数据库与 SQL 语言

结构化查询语言包含6个部分：

- 1、数据查询语言（**DQL**: Data Query Language）：其语句，也称为“数据检索语句”，用以从表中获得数据，确定数据怎样在应用程序给出。保留字 SELECT 是 DQL (也是所有 SQL) 用得最多的动词，其他 DQL 常用的保留字有 WHERE , ORDER BY , GROUP BY 和 HAVING。这些 DQL 保留字常与其它类型的 SQL 语句一起使用。
- 2、数据操作语言（**DML** : Data Manipulation Language）：其语句包括动词 INSERT、UPDATE 和 DELETE。它们分别用于添加、修改和删除。
- 3、事务控制语言（**TCL**）：它的语句能确保被 DML 语句影响的表的所有行及时得以更新。包括 COMMIT (提交) 命令、SAVEPOINT (保存点) 命令、ROLLBACK (回滚) 命令。
- 4、数据控制语言（**DCL**）：它的语句通过 GRANT 或 REVOKE 实现权限控制，确定单个用户和用户组对数据库对象的访问。某些 RDBMS 可用 GRANT 或 REVOKE 控制对表单个列的访问。
- 5、数据定义语言（**DDL**）：其语句包括动词 CREATE,ALTER 和 DROP。在数据库中创建新表或修改、删除表（CREATE TABLE 或 DROP TABLE）；为表加入索引等。
- 6、指针控制语言（**CCL**）：它的语句，像 DECLARE CURSOR , FETCH INTO 和 UPDATE WHERE CURRENT 用于对一个或多个表单独行的操作。

二、关系数据库与 SQL 语言

SQL 的各个版本:

1986年 , ANSI X3.135-1986 , ISO/IEC 9075:1986 , SQL-86

1989年 , ANSI X3.135-1989 , ISO/IEC 9075:1989 , SQL-89

1992年 , ANSI X3.135-1992 , ISO/IEC 9075:1992 , SQL-92 (SQL2)

1999年 , ISO/IEC 9075:1999 , SQL:1999 (SQL3)

2003年 , ISO/IEC 9075:2003 , SQL:2003

2008年 , ISO/IEC 9075:2008 , SQL:2008

2011年 , ISO/IEC 9075:2011 , SQL:2011

SQL 解析技术 : 手写 , yacc , antlr

三、MySQL 数据库

瑞典的 MySQL AB 创立于1995年。

2008年1月16日 MySQL AB 被 Sun Microsystems 收购。

2009年4月20日，甲骨文 (Oracle) 收购 Sun Microsystems 公司。

其后分离成两个版本：MySQL、MariaDB



三、MySQL 数据库

MySQL 的版本

- 4.0 支持 InnoDB , 事务
- 2003年 , 5.0
- 5.6 ==> 历史使用最多的版本
- 5.7 ==> 近期使用最多的版本
- 8.0 ==> 最新和功能完善的版本

思考：选择学习哪个版本？

三、MySQL 数据库

- 5.6/5.7的差异

5.7支持：

- 多主
- MGR 高可用
- 分区表
- json
- 性能
- 修复 XA 等

三、MySQL 数据库

- 5.7/8.0的差异
- 通用表达式
- 窗口函数
- 持久化参数
- 自增列持久化
- 默认编码 utf8mb4
- DDL 原子性
- JSON 增强
- 不再对 group by 进行隐式排序 ? ? ==> 坑

三、MySQL 数据库

MySQL 安装方式：

- 1、压缩包：zip
- 2、安装文件与安装命令：installer.exe/choco , yum/apt , brew
- 3、docker pull mysql:5.7.32

MySQL 常用命令演示

三、MySQL 数据库

MySQL 核心功能：

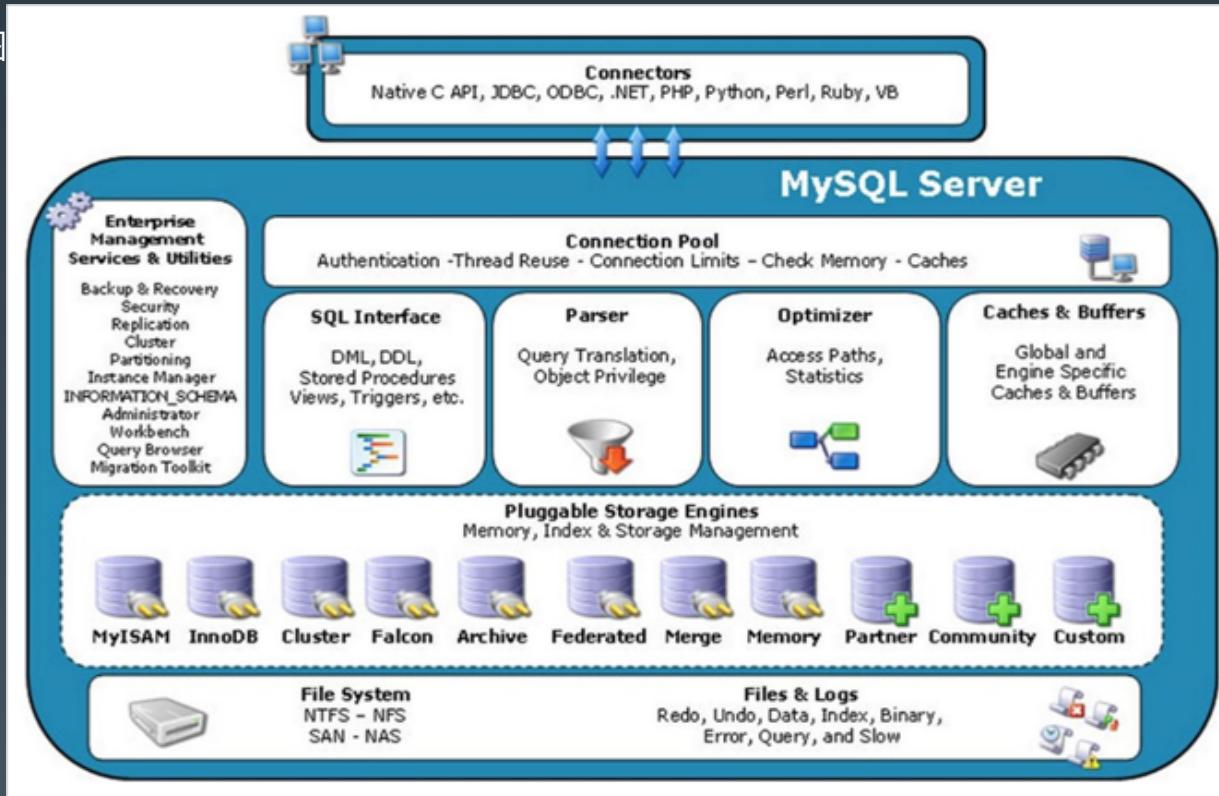
- 1、数据库引擎
- 2、管理工具
- 3、任何平台环境
- 4、数据库生态
- 5、一致性、成熟度、性能、可维护性、高可用与稳定性

MySQL 与 Oracle、DB2、SQLServer 对比。

思考：为什么大家要去 IOE？

四、深入数据库原理

MySQL 架构图



四、深入数据库原理

MySQL 文件存储结构

独占模式

- 1) 、日志组文件 : ib_logfile0 和 ib_logfile1 , 默认均为5M
- 2) 、表结构文件 : *.frm
- 3) 、独占表空间文件 : *.ibd
- 4) 、字符集和排序规则文件 : db.opt
- 5) 、binlog 二进制日志文件 : 记录主数据库服务器的 DDL 和 DML 操作
- 6) 、二进制日志索引文件 : master-bin.index

共享模式 innodb_file_per_table=1

- 1) 、数据都在 ibdata1

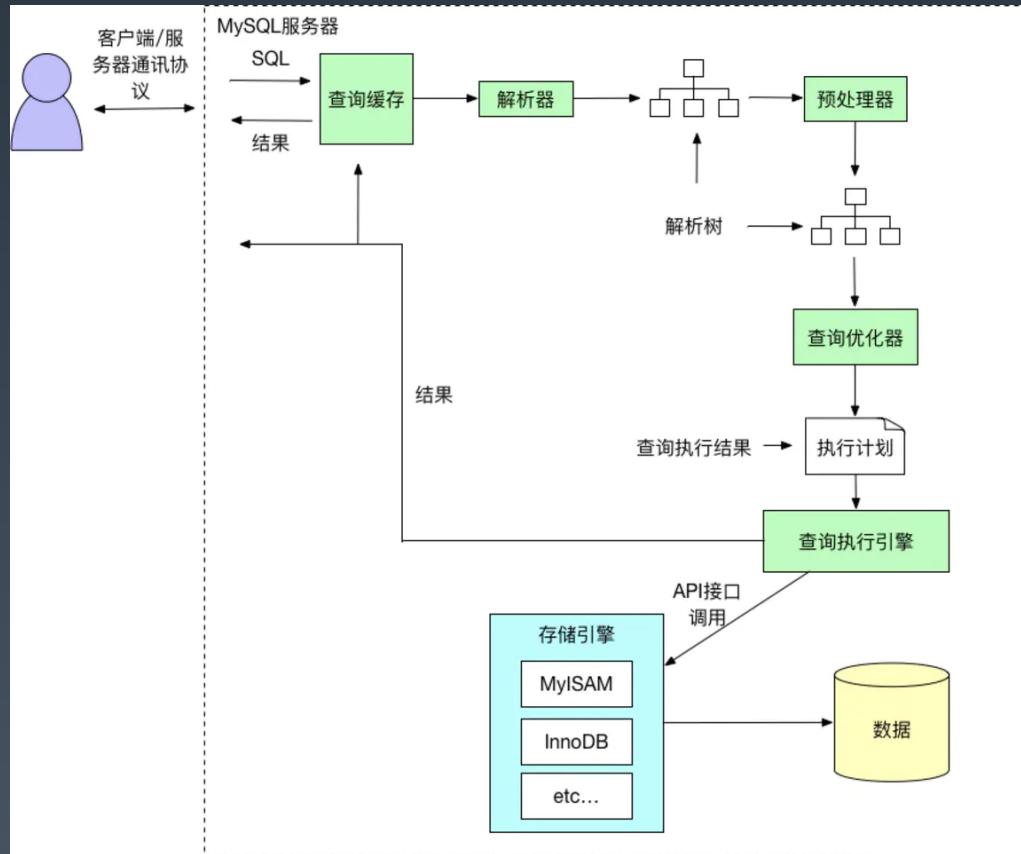
四、深入数据库原理

演示 MySQL 的文件结构

演示 MySQL 默认的数据库结构和表

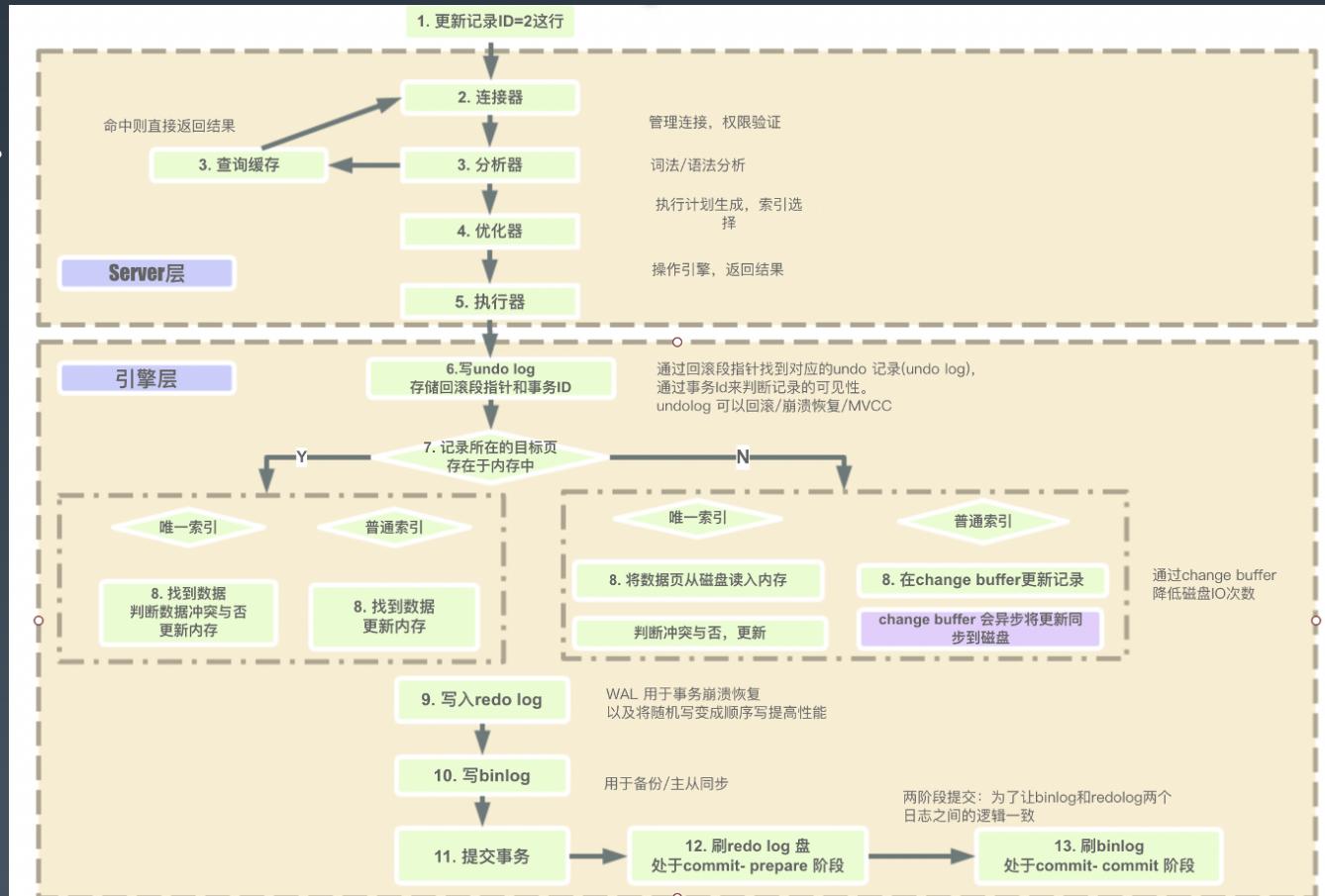
四、深入数据库原理

MySQL 简化执行流程。



四、深入数据库原理

MySQL 详细执行流程。



四、深入数据库原理

MySQL 执行引擎和状态

存储引擎	myisam	innodb	memory	archive
存储限制	256TB	64TB	有	无
事务	-	支持	-	-
索引	支持	支持	支持	-
锁的粒度	表锁	行锁	表锁	行锁
数据压缩	支持	-	-	支持
外键	-	支持	-	-

状态	表示的含义
sleep	线程正在等待客户端发送新的请求
query	线程正在执行查询或者正在讲结果发送给客户端
locked	mysql服务器层,线程正在等待表锁
analyzing and statistics	线程正在收集存储引擎的统计信息,并生成查询的执行计划
copying to temp table	线程正在执行查询,并且将结果复制到临时表中,一般是做group by操作,要么是文件排序,或者union
on disk	将内存中的临时表放在磁盘上
sorting result	线程正在对结果集进行排序
sending data	线程在多个状态之间传递数据或者生成结果集或者向客户端返回数据

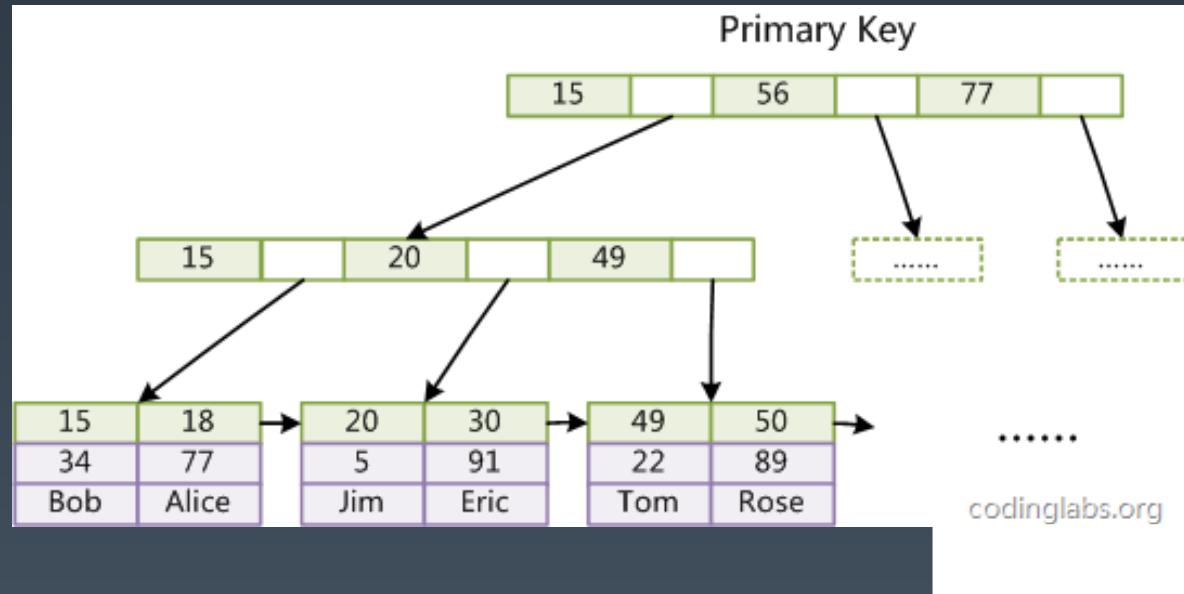
四、深入数据库原理

SQL 执行顺序

实际上这个过程也并不是绝对这样的，
中间 mysql 会有部分的优化以达到最佳的优化效果，
比如在 select 筛选出找到的数据集



四、深入数据库原理



数据是按页来分块的，
当一个数据被用到时，其附近的数据也通常
会马上被使用。

InnoDB 使用 B+ 树实现聚集索引。

思考：为什么单表数据一般不超过2000万行？

四、深入数据库原理

计算一下单表数据容量。

四、深入数据库原理

操作演示。

使用 IDE 工具访问和操作。

数据导入导出等。

五、MySQL 配置优化经验

查看参数配置

- show variables like xxx
- select @@last_insert_id;

my.cnf 文件

[mysqld]

server

[mysql]

client

五、MySQL 配置优化经验

1) 连接请求的变量

- 1、max_connections
- 2、back_log
- 3、wait_timeout和interative_timeout

五、MySQL 配置优化经验

2) 缓冲区变量

4、key_buffer_size

5、query_cache_size (查询缓存简称 QC)

6、max_connect_errors :

7、sort_buffer_size :

8、max_allowed_packet=32M

9、join_buffer_size=2M

10、thread_cache_size=300

五、MySQL 配置优化经验

3) 配置 Innodb 的几个变量

- 11、innodb_buffer_pool_size
- 12、innodb_flush_log_at_trx_commit
- 13、innodb_thread_concurrency=0
- 14、innodb_log_buffer_size
- 15、innodb_log_file_size=50M
- 16、innodb_log_files_in_group=3
- 17、read_buffer_size=1M
- 18、read_rnd_buffer_size=16M
- 19、bulk_insert_buffer_size=64M
- 20、binary log

六、数据库设计优化经验

- 1.如何恰当选择引擎？
- 2.库表如何命名？
- 3.如何合理拆分宽表？
- 4.如何选择恰当数据类型：明确、尽量小
 - char、varchar 的选择
 - (text/blob/clob) 的使用问题？
 - 文件、图片是否要存入到数据库？
 - 时间日期的存储问题？
 - 数值的精度问题？
- 5.是否使用外键、触发器？

六、数据库设计优化经验

- 6.唯一约束和索引的关系？
- 7.是否可以冗余字段？
- 8.是否使用游标、变量、视图、自定义函数、存储过程？
- 9.自增主键的使用问题？
- 10.能够在线修改表结构（DDL 操作）？
- 11.逻辑删除还是物理删除？
- 12.要不要加 create_time/update_time 时间戳？
- 13.数据库碎片问题？
- 14.如何快速导入导出、备份数据？

七、MySQL 事务原理

ACID (1页)

几种不同的锁。 (1-2页) IS GAP nextkey

隔离级别 , 解决什么问题 (脏读、不可重复度、幻读) (3-4页)

undo、 redo log 机制。

MySQL 的 MVCC 机制 (2页)

大概写8-12页。

参考 : <https://www.zhihu.com/collection/361093857> 这里很多 , 找一找

undo redo : <https://zhuanlan.zhihu.com/p/190886874>

MVCC : <https://www.zhihu.com/question/334408495/answer/881317999>

ACID 隔离级别 : <https://www.zhihu.com/column/distributedDatabase> 这里有两篇

七、MySQL 事务原理

事务可靠性模型 ACID:

- Atomicity: 原子性, 一次事务中的操作要么全部成功, 要么全部失败。
- Consistency: 一致性, 跨表、跨行、跨事务, 数据库始终保持一致状态。
- Isolation: 隔离性, 可见性, 保护事务不会互相干扰, 包含4种隔离级别。
- Durability: 持久性, 事务提交成功后, 不会丢数据。如电源故障, 系统崩溃。

性能 vs. 可靠性

InnoDB:

双写缓冲区、故障恢复、操作系统、fsync()、磁盘存储、缓存、UPS、网络、备份策略

七、MySQL 事务原理

表级锁

意向锁: 表明事务稍后要进行哪种类型的锁定

- 共享意向锁(IS): 打算在某些行上设置共享锁
- 排他意向锁(IX): 打算对某些行设置排他锁
- Insert意向锁: Insert操作设置的间隙锁

其他

- 自增锁(AUTO-INC)
- LOCK TABLES/DDL

• 共享锁(S)

• 排他锁(X)

上锁前需要先上意向锁!

锁类型的兼容性

	X	IX	S	IS
X	冲突	冲突	冲突	冲突
IX	冲突	兼容	冲突	兼容
S	冲突	冲突	兼容	兼容
IS	冲突	兼容	兼容	兼容

SHOW ENGINE INNODB STATUS;

七、MySQL 事务原理

行级锁(InnoDB)

- 记录锁(Record): 始终锁定索引记录，注意隐藏的聚簇索引;
- 间隙锁(Gap):
- 临键锁(Next-Key): 记录锁+间隙锁的组合; 可“锁定”表中不存在记录
- 谓词锁(Predicate): 空间索引

• 共享锁(S)

• 排他锁(X)

死锁:

- 阻塞与互相等待
- 增删改、锁定读
- 死锁检测与自动回滚
- 锁粒度与程序设计

七、MySQL 事务原理

《SQL:1992标准》规定了四种事务隔离级别(Isolation):

- 读未提交: READ UNCOMMITTED
- 读已提交: READ COMMITTED
- 可重复读: REPEATABLE READ
- 可串行化: SERIALIZABLE

事务隔离是数据库的基础特征。

隔离级别	并发性 可靠性 一致性 可重复性
------	---------------------------

MySQL:

- 可以设置全局的默认隔离级别
- 可以单独设置会话的隔离级别
- InnoDB 实现与标准之间的差异

七、MySQL 事务原理

读未提交: READ UNCOMMITTED

- 很少使用
- 不能保证一致性
- 脏读(**dirty read**) : 使用到从未被确认的数据(例如: 早期版本、回滚)

锁:

- 以非锁定方式执行
- 可能的问题: 脏读、幻读、不可重复读

七、MySQL 事务原理

读已提交: READ COMMITTED

- 每次查询都会设置和读取自己的新快照。
- 仅支持基于行的 bin-log
- UPDATE 优化: 半一致读(semi-consistent read)
- 不可重复读: 不加锁的情况下, 其他事务 UPDATE 或 DELETE 会对查询结果有影响
- 幻读(Phantom): 加锁后, 不锁定间隙, 其他事务可以 INSERT。

锁:

- 锁定索引记录, 而不锁定记录之间的间隙
- 可能的问题: 幻读、不可重复读

七、MySQL 事务原理

可重复读: REPEATABLE READ

- InnoDB 的默认隔离级别
- 使用事务第一次读取时创建的快照
- 多版本技术

锁:

- 使用唯一索引的唯一查询条件时, 只锁定查找到的索引记录, 不锁定间隙。
- 其他查询条件, 会锁定扫描到的索引范围, 通过间隙锁或临键锁来阻止其他会话在这个范围内插入值。
- 可能的问题: InnoDB 不能保证没有幻读, 需要加锁

七、MySQL 事务原理

串行化: SERIALIZABLE

最严格的级别, 事务串行执行, 资源消耗最大;

问题回顾:

- 脏读(**dirty read**): 使用到从未被确认的数据(例如: 早期版本、回滚)
- 不可重复读: 不加锁的情况下, 其他事务 `update` 或 `delete` 会对结果集有影响
- 幻读(**Phantom**): 加锁之后, 相同的查询语句, 在不同的时间点执行时, 产生不同的结果集

怎么解决?

提高隔离级别、使用间隙锁或临键锁

七、MySQL 事务原理

undo log: 撤消日志

- 保证事务的原子性
- 用处: 事务回滚, 一致性读、崩溃恢复。
- 记录事务回滚时所需的撤消操作
- 一条 INSERT 语句, 对应一条 DELETE 的 undo log
- 每个 UPDATE 语句, 对应一条相反 UPDATE 的 undo log

保存位置:

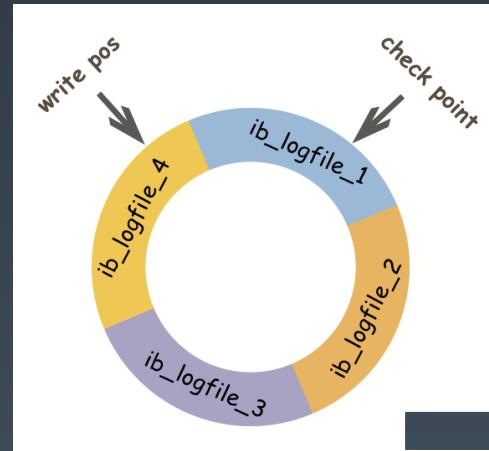
- system tablespace (MySQL 5.7默认)
- undo tablespaces (MySQL 8.0默认)

回滚段(rollback segment)

七、MySQL 事务原理

redo log: 重做日志

- 确保事务的持久性，防止事务提交后数据未刷新到磁盘就掉电或崩溃。
- 事务执行过程中写入 redo log，记录事务对数据页做了哪些修改。
- 提升性能: WAL(Write-Ahead Logging) 技术, 先写日志, 再写磁盘。
- 日志文件: ib_logfile0, ib_logfile1
- 日志缓冲: innodb_log_buffer_size
- 强刷: fsync()



七、MySQL 事务原理

MVCC: 多版本并发控制, multiversion concurrency control

- 使 InnoDB 支持一致性读: READ COMMITTED 和 REPEATABLE READ。
- 让查询不被阻塞、无需等待被其他事务持有的锁，这种技术手段可以增加并发性能。
- InnoDB 保留被修改行的旧版本。
- 查询正在被其他事务更新的数据时，会读取更新之前的版本。
- 每行数据都存在一个版本号, 每次更新时都更新该版本
- 这种技术在数据库领域的使用并不普遍。某些数据库, 以及某些 MySQL 存储引擎都不支持。

聚簇索引的更新 = 替换更新

二级索引的更新 = 删除+新建

七、MySQL 事务原理

MVCC 实现机制

- 隐藏列
- 事务链表，保存还未提交的事务，事务提交则会从链表中摘除
- Read view: 每个 SQL 一个, 包括 rw trx_ids, low_limit_id, up_limit_id, low_limit_no 等
- 回滚段: 通过 undo log 动态构建旧版本数据

隐藏列:	DB_TRX_ID	DB_ROLL_PTR	DB_ROW_ID
长度:	6-byte	7-byte	6-byte
说明:	指示最后插入或更新该行的事务ID	回滚指针。指向回滚段中写入的 undo log 记录	聚簇row ID/聚簇索引

八、MySQL 对 XA 的支持

1、X/Open DTP 模型与 XA 规范

X/Open，即现在的 open group，是一个独立的组织，主要负责制定各种行业技术标准。X/Open 组织主要由各大知名公司或者厂商进行支持，这些组织不光遵循 X/Open 组织定义的行业技术标准，也参与到标准的制定。

The Open Group Platinum Members

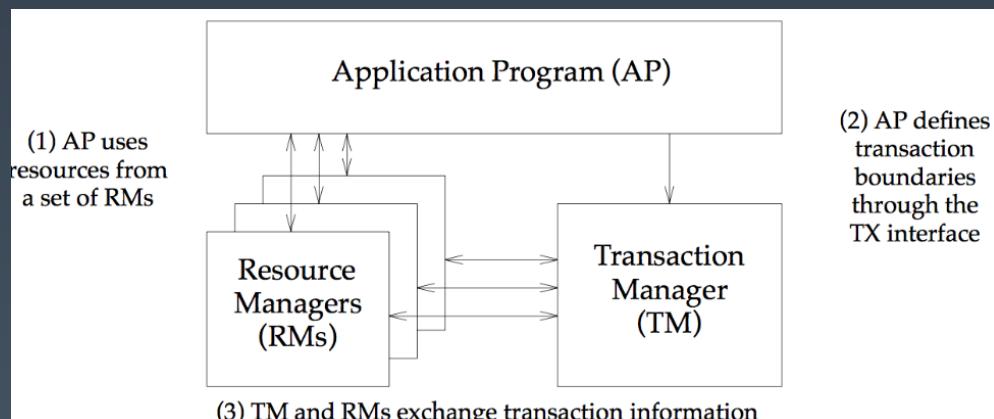


All Open Group Members

应用程序(Application Program，简称 AP)：用于定义事务边界(即定义事务的开始和结束)，并且在事务边界内对资源进行操作。

资源管理器(Resource Manager，简称 RM)：如数据库、文件系统等，并提供访问资源的方式

事务管理器(Transaction Manager，简称 TM)：负责分配事务唯一标识，监控事务的执行进度，并负责事务的提交、回滚等。



八、MySQL 对 XA 的支持

XA 接口

`xa_start` : 负责开启或者恢复一个事务分支

`xa_end` : 负责取消当前线程与事务分支的关联

`xa_prepare` : 询问 RM 是否准备好了提交事务分支

`xa_commit` : 通知 RM 提交事务分支

`xa_rollback` : 通知 RM 回滚事务分支

`xa_recover` : 需要恢复的 XA 事务

Name	Description
<code>ax_reg</code>	Register an RM with a TM.
<code>ax_unreg</code>	Unregister an RM with a TM.
<code>xa_close</code>	Terminate the AP's use of an RM.
<code>xa_commit</code>	Tell the RM to commit a transaction branch.
<code>xa_complete</code>	Test an asynchronous <code>xa_</code> operation for completion.
<code>xa_end</code>	Dissociate the thread from a transaction branch.
<code>xa_forget</code>	Permit the RM to discard its knowledge of a heuristically-completed transaction branch.
<code>xa_open</code>	Initialise an RM for use by an AP.
<code>xa_prepare</code>	Ask the RM to prepare to commit a transaction branch.
<code>xa_recover</code>	Get a list of XIDs the RM has prepared or heuristically completed.
<code>xa_rollback</code>	Tell the RM to roll back a transaction branch.
<code>xa_start</code>	Start or resume a transaction branch - associate an XID with future work that the thread requests of the RM.

八、MySQL 对 XA 的支持

MySQL 从5.0.3开始支持 XA 分布式事务，且只有 InnoDB 存储引擎支持。MySQL Connector/J 从5.0.0版本开始支持

Engine	Support	Comment	Transactions		
			XA	Savepoints	
InnoDB	DEFAULT	Supports transactions, row-level locking, and foreign keys	YES	YES	YES
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
MEMORY	YES	Hash based, stored in memory, useful for temporary tables	NO	NO	NO
BLACKHOLE	YES	/dev/null storage engine (anything you write to it disappears)	NO	NO	NO
MyISAM	YES	MyISAM storage engine	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
ARCHIVE	YES	Archive storage engine	NO	NO	NO
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO
FEDERATED	NO	Federated MySQL storage engine	NULL	NULL	NULL

在 DTP 模型中，MySQL 属于资源管理器(RM)。而完整的分布式事务中，存在多个 RM，由事务管理器 TM 来统一进行协调。

MySQL XA 事务 SQL 语法

```

XA {START|BEGIN} xid [JOIN|RESUME] //开启XA事务，如果使用的是XA START而不是XA BEGIN，那么不支持[JOIN|RESUME]，xid是一个唯一值，表示事务分支标识
XA END xid [SUSPEND [FOR MIGRATE]] //结束一个XA事务，不支持[SUSPEND [FOR MIGRATE]]
XA PREPARE xid 准备提交
XA COMMIT xid [ONE PHASE] //提交，如果使用了ONE PHASE，则表示使用一阶段提交。两阶段提交协议中，如果只有一个RM参与，那么可以优化为一阶段提交
XA ROLLBACK xid //回滚
XA RECOVER [CONVERT XID] //列出所有处于PREPARE阶段的XA事务
    
```

八、MySQL 对 XA 的支持

MySQL 中使用 xid 来作为一个事务分支的标识符。事实上 xid 作为事务分支标识符是在 XA 规范中定义的，在<< Distributed Transaction Processing: The XA Specification>> 4.2 节中，规定了一个 xid 的结构，通过 C 语言进行描述，如下：

```
#define XIDDATASIZE 128 /* size in bytes */
#define MAXGTRIDSIZE 64 /* maximum size in bytes of gtrid */
#define MAXBQUALSIZE 64 /* maximum size in bytes of bqual */
struct xid_t {
    long formatID; /* format identifier */
    long gtrid_length; /* value 1-64 */
    long bqual_length; /* value 1-64 */
    char data[XIDDATASIZE];
};

/*
 * A value of -1 in formatID means that the XID is null.
*/
typedef struct xid_t XID;
/*
 * Declarations of routines by which RMs call TMs:
*/
extern int ax_reg(int, XID *, long);
extern int ax_unreg(int, long);
```

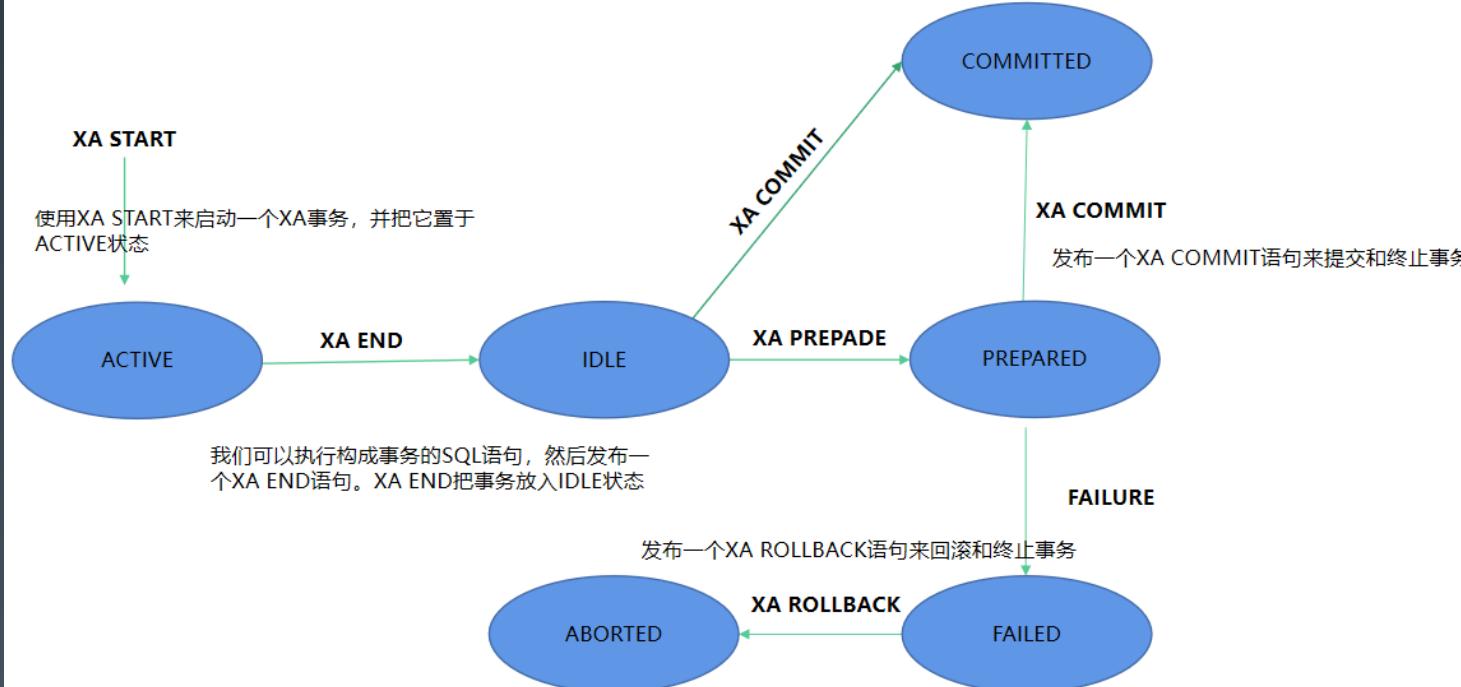
```
mysql> XA RECOVER;
+-----+-----+-----+-----+
| formatID | gtrid_length | bqual_length | data          |
+-----+-----+-----+-----+
|      1 |         6 |         6 | g12345b67890 |
```

- gtrid : 全局事务标识符(global transaction identifier)，最大不能超过64字节
- bqual : 分支限定符(branch qualifier)，最大不能超过64字节
- formatId : 记录 gtrid、bqual 的格式，类似于 memcached 中 flags 字段的作用。XA 规范中通过一个结构体约定了 xid 的组成部分，但是并没有规定 data 中存储的 gtrid、bqual 内容到底应该是什么格式
- data : xid 的值，其是 gtrid 和 bqual 拼接后的内容。因为 gtrid 和 bqual 最大都是64个字节，因此 data 的最大长度为 128。

八、MySQL 对 XA 的支持

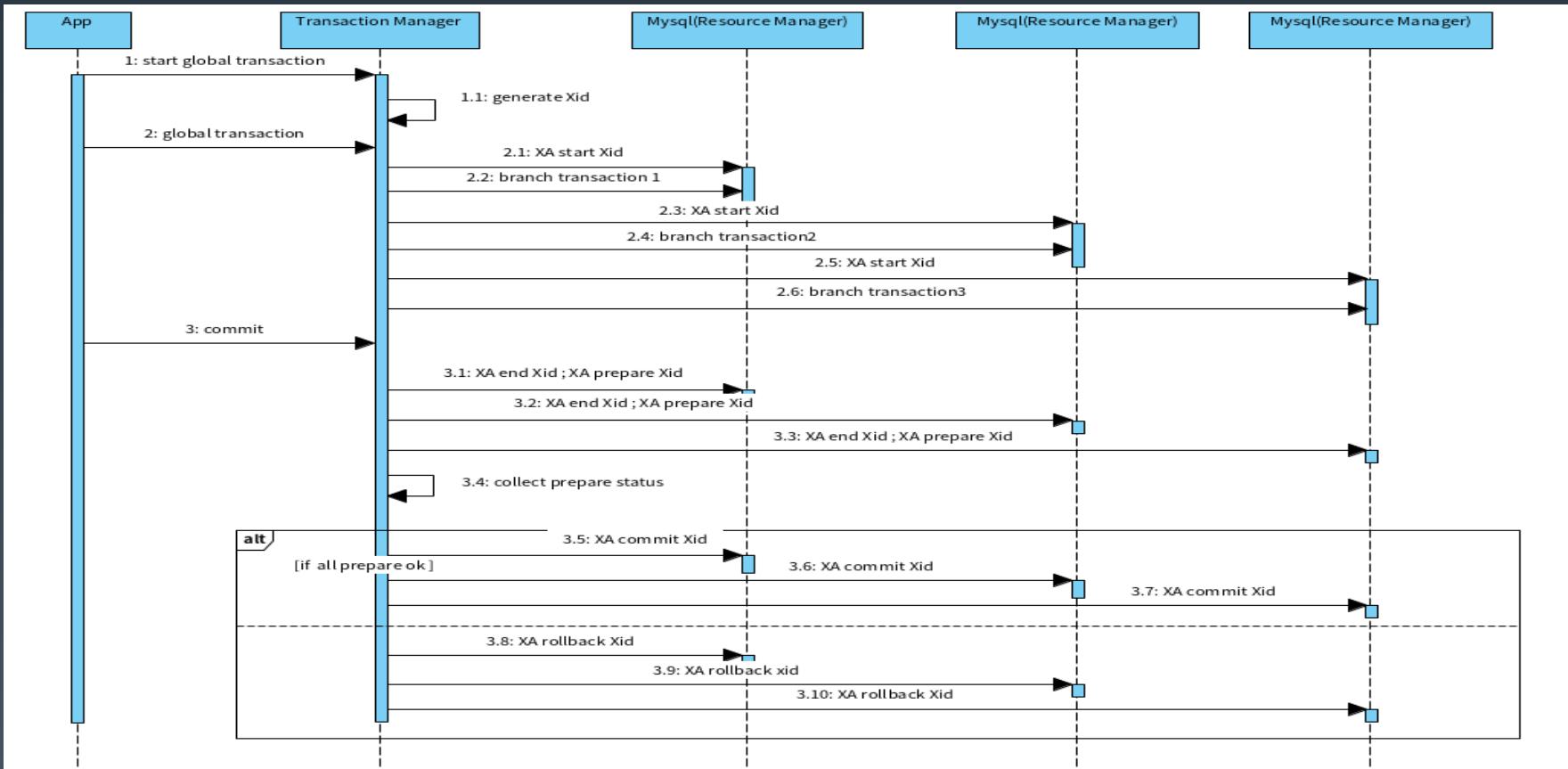
ACTIVE状态的 XA 事务，我们可以执行构成事务的SQL语句，然后发布一个XA END语句。XA END把事务放入IDLE状态

MySQL XA 事务状态

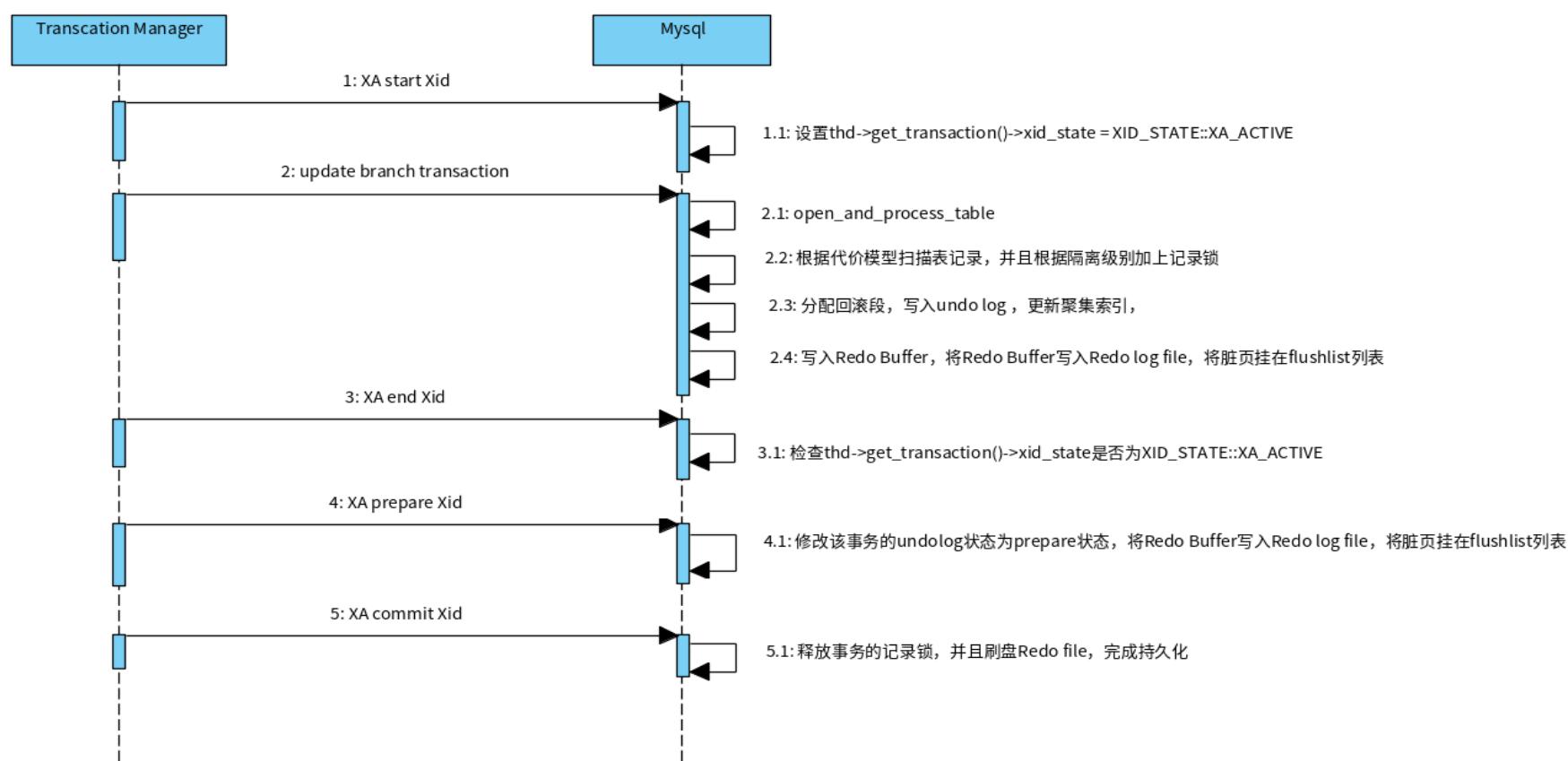


XA 事务和非 XA 事务(即本地事务)是互斥的。例如，已经执行了“XA START”命令来开启一个 XA 事务，则本地事务不会被启动，直到 XA 事务已经被提交或被回滚为止。相反的，如果已经使用 START TRANSACTION 启动一个本地事务，则 XA 语句不能被使用，直到该事务被提交或被回滚为止。

八、MySQL 对 XA 的支持



八、MySQL 对 XA 的支持



思考一：持久化事务协调阶段的各个状态

TM作为一个单点的事务协同器，很有可能宕机，出现单点故障。其本身职责主要是事务协调，属于无状态的服务。宕机重启后，可根据持久化的全局事务状态来恢复TM的执行逻辑，所以，需要将阶段的各个协调阶段以及该阶段中每个RM的执行状态持久化到独立DB中，多个TM共享一个持久化DB。具体的阶段有，prepare阶段的子阶段有branch_transaction_send、prepare_send、prepare_ack阶段，commit阶段的子阶段有commit_send、commit_ack阶段，记录每个子阶段每个RM的执行状态

思考二：是否需要并行发送语句

branch_transaction_send、prepare_send、commit_send阶段，如果TM往RM发送语句是串行执行的，单个global transaction 执行时间加长，TM的TPS（每秒事务请求数）会降低，可以在这些阶段将已生成的语句，通过线程池并行发送到各个RM，TM同时同等待语句的返回值，延时大为降低。

思考三：TM在prepare_send阶段前宕机，重启恢复后，继续执行prepare_send动作

思考四：TM在prepare_send阶段时宕机，可能会有部分RM收到prepare语句，部份没有收到，重启后，往收到prepare语句的RM发送rollback语句

思考五：TM在prepare_ack阶段记录完各个RM的执行状态后宕机，重启后，根据日志状态发起rollback或者commit语句。

思考六：TM在commit_send阶段时宕机，可能会有部分RM收到commit语句，部份没有收到，重启后，往没有收到commit语句的RM发送commit语句。

思考七：TM在commit_ack阶段记录完各个RM的执行状态后宕机，重启后，根据日志状态发起重试commit语句或者不操作。

思考八：RM超长时间没有收到TM的rollback或者commit语句，一直持有记录锁，RM要有自动rollback或者commit的功能。

八、MySQL 对 XA 的支持

MySQL各版本对XA的支持与优化

MySQL < 5.7 版本会出现的问题（后续我们可以来重现）

- 已经prepare的事务，在客户端退出或者服务宕机的时候，2PC的事务会被回滚。
- 在服务器故障重启提交后，相应的Binlog被丢失

MySQL 5.6版本在客户端退出的时候，自动把已经prepare的事务回滚了，那么MySQL为什么要这样做？这主要取决于MySQL的内部实现，MySQL 5.7以前的版本，对于prepare的事务，MySQL是不会记录binlog的（官方说是减少fsync，起到了优化的作用）。只有当分布式事务提交的时候才会把前面的操作写入binlog信息，所以对于binlog来说，分布式事务与普通的事务没有区别，而prepare以前的操作信息都保存在连接的IO_CACHE中，如果这个时候客户端退出了，以前的binlog信息都会被丢失，再次重连后允许提交的话，会造成Binlog丢失，从而造成主从数据的不一致，所以官方在客户端退出的时候直接把已经prepare的事务都回滚了！

MySQL > 5.7 版本的优化 (<https://dev.mysql.com/worklog/task/?id=6860>)

MySQL对于分布式事务，在prepare的时候就完成了写Binlog的操作，通过新增一种叫XA_prepare_log_event的event类型来实现，这是与以前版本的主要区别(以前版本prepare时不写Binlog)。！

八、MySQL 对 XA 的支持

XA协议存在的问题

1. 同步阻塞问题

全局事务内部包含了多个独立的事务分支，这一组事务分支要不都成功，要不都失败。各个事务分支的ACID特性共同构成了全局事务的ACID特性。也就是将单个事务分支的支持的ACID特性提升一个层次(up a level)到分布式事务的范畴。即使在非分布事务中(即本地事务)，如果对操作读很敏感，我们也需要将事务隔离级别设置为SERIALIZABLE。而对于分布式事务来说，更是如此，可重复读隔离级别不足以保证分布式事务一致性。也就是说，如果我们使用mysql来支持XA分布式事务的话，那么最好将事务隔离级别设置为SERIALIZABLE。地球人都知道，SERIALIZABLE(串行化)是四个事务隔离级别中最高的一个级别，也是执行效率最低的一个级别

2. 单点故障

由于协调者的重要性，一旦协调者TM发生故障。参与者RM会一直阻塞下去。尤其在第二阶段，协调者发生故障，那么所有的参与者还都处于锁定事务资源的状态中，而无法继续完成事务操作。(如果是协调者挂掉，可以重新选举一个协调者，但是无法解决因为协调者宕机导致的参与者处于阻塞状态的问题)

3. 数据不一致

在二阶段提交的阶段二中，当协调者向参与者发送commit请求之后，发生了局部网络异常或者在发送commit请求过程中协调者发生了故障，这回导致只有一部分参与者接受到了commit请求。而在这部分参与者接到commit请求之后就会执行commit操作。但是其他部分未接到commit请求的机器则无法执行事务提交。于是整个分布式系统便出现了数据不一致性的现象。

八、MySQL 对 XA 的支持

主流的开源XA分布式事务解决方案



ATOMIKOS

- TM: 去中心化设计, 性能较高
- 日志存储件: 只支持文件
- 扩展性: 较好
- 事务恢复: 只支持单机事务恢复
- 标准的XA实现



- TM: 去中心化设计, 性能较高
- 日志存储件: 支持文件, 数据库
- 扩展性: 一般
- 事务恢复: 支持集群模式恢复
- 标准的XA实现



- TM: 中心化设计, 性能较差, BUG多
- 日志存储件: 支持文件, 数据库
- 扩展性: 一般
- 事务恢复: 问题很多, 未能正确恢复
- 非标准的XA实现

THANKS! |  极客大学