



去哪儿应届生训练营  
Qunar Fresh Camp

# MySQL数据库培训

刘鹏飞

To be coding, to be working!

**刘鹏飞**

pengfeis.liu

技术运营中心/数据库/DBA

目前主要负责MySQL数据库的运维和自动化相关工作;

具有多年的数据库运维工作经验;



# MySQL数据库简介

排名	上月	数据库名称	数据库类型	分数	变化
1	1	Oracle	Relational	1287.74	📈 +24.92
2	2	MySQL	Relational	1189.21	📉 -12.89
3	3	Microsoft SQL Server	Relational	933.83	📉 -7.37
4	4	PostgreSQL	Relational	620.84	📈 +5.55
5	5	MongoDB	Document	480.73	📈 +2.49
6	7	Redis	Key-value	175.31	📉 -3.71
7	6	IBM Db2	Relational	159.19	📉 -1.14
8	8	Elasticsearch	Search engine	156.0	📉 -1.7
9	10	Microsoft Access	Relational	141.82	📉 -1.62
10	9	SQLite	Relational	135.44	📈 +0.7
11	11	Cassandra	Wide column	115.45	📉 -2.56
12	12	MariaDB	Relational	111.58	📈 +0.45

各大互联网公司和证券，银行等行业公司都在大量的使用MySQL数据库；

具有免费，开源，高性能等特点；

重要里程碑：

mysql-5.5.5开始,InnoDB作为默认存储引擎，InnoDB作为支持事务的存储引擎，拥有相关的RDBMS特性：包括ACID事务支持，参考完整性(外键)，灾难恢复能力等特性。

去哪儿网目前核心的系统基本都在MySQL上，有约2000+实例

# 关于线上课程的说明

- 目前去哪儿网公司MySQL数据库版本以5.6和5.7为主
- 数据查询语言DQL，数据操纵语言DML语句一定要有where条件
- DDL操作中**drop 和truncate表非常危险，另外不要删除表的列**
- 表的存储引擎统一使用InnoDB，不允许MyISAM存储引擎
- 表字符集统一使用utf8mb4
- 权限申请：需要什么权限就申请什么权限，够用为宜，申请的权限不要轻易回收
- **所有数据库操作一定要谨慎，多次确认！尤其是涉及到批量修改和删除数据时，一定要找组内人员确认！**

# 课程目标

- 了解MySQL集群架构及其特点
- 了解MySQL表和索引结构
- 熟悉MySQL表和索引使用规范
- 熟悉MySQL的执行计划
- 熟悉MySQL自动化运维平台

# 目录

- 1、Qunar MySQL集群架构介绍
- 2、MySQL数据库使用规范
- 3、自动化运维平台使用介绍



1

# MySQL集群架构介绍

# MySQL集群架构介绍

## PXC架构

是一种节点对等的，multi-master架构；  
哨兵管理；  
连接方式使用namespace，支持读写分离。

## 3M架构

数据库是主从架构；  
使用3M monitor和agent管理；  
连接方式使用VIP连接，支持读写分离。

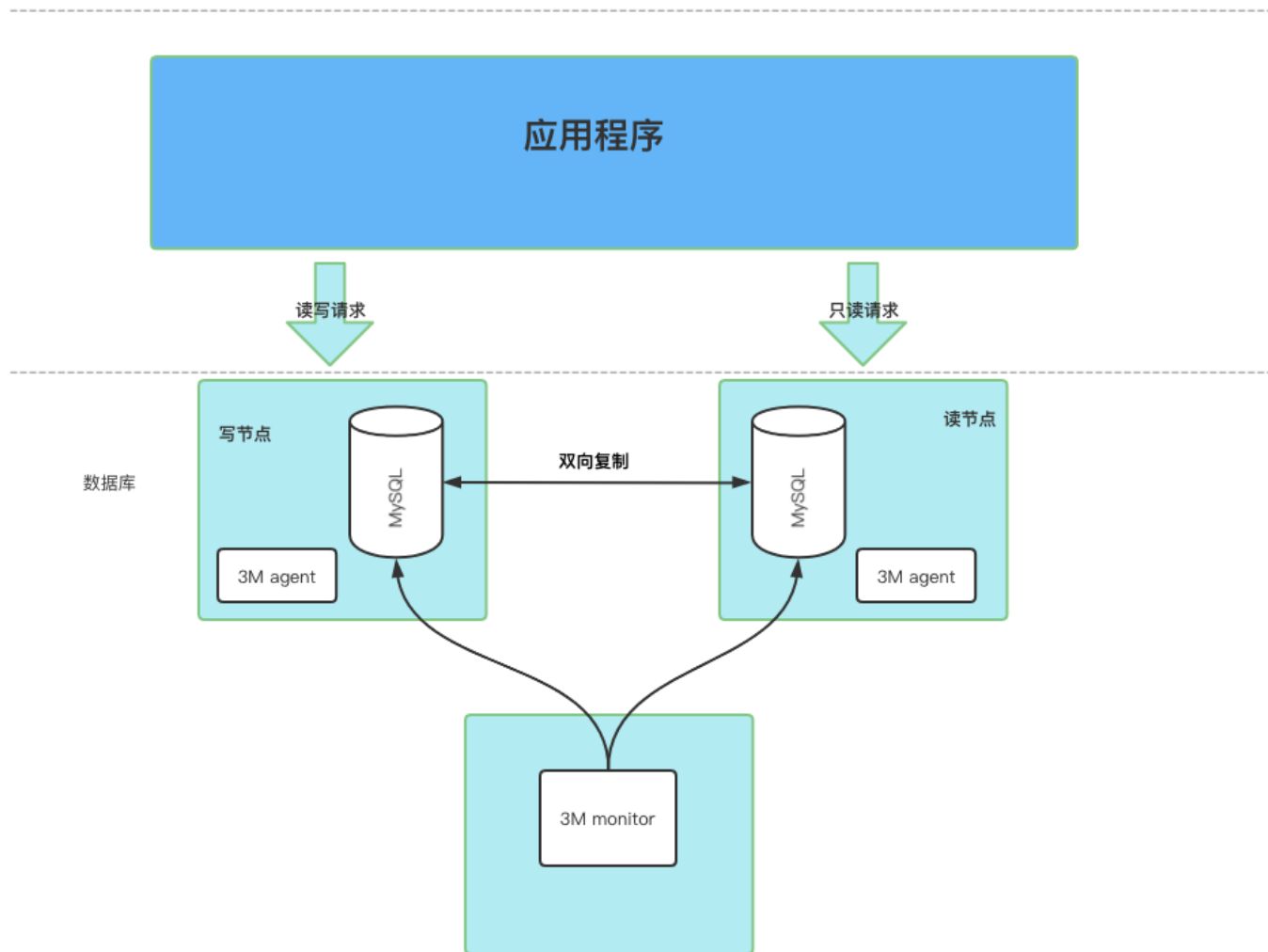


## QMHA架构

数据库是主从结构；  
哨兵管理；  
连接方式使用namespace，支持读写分离。



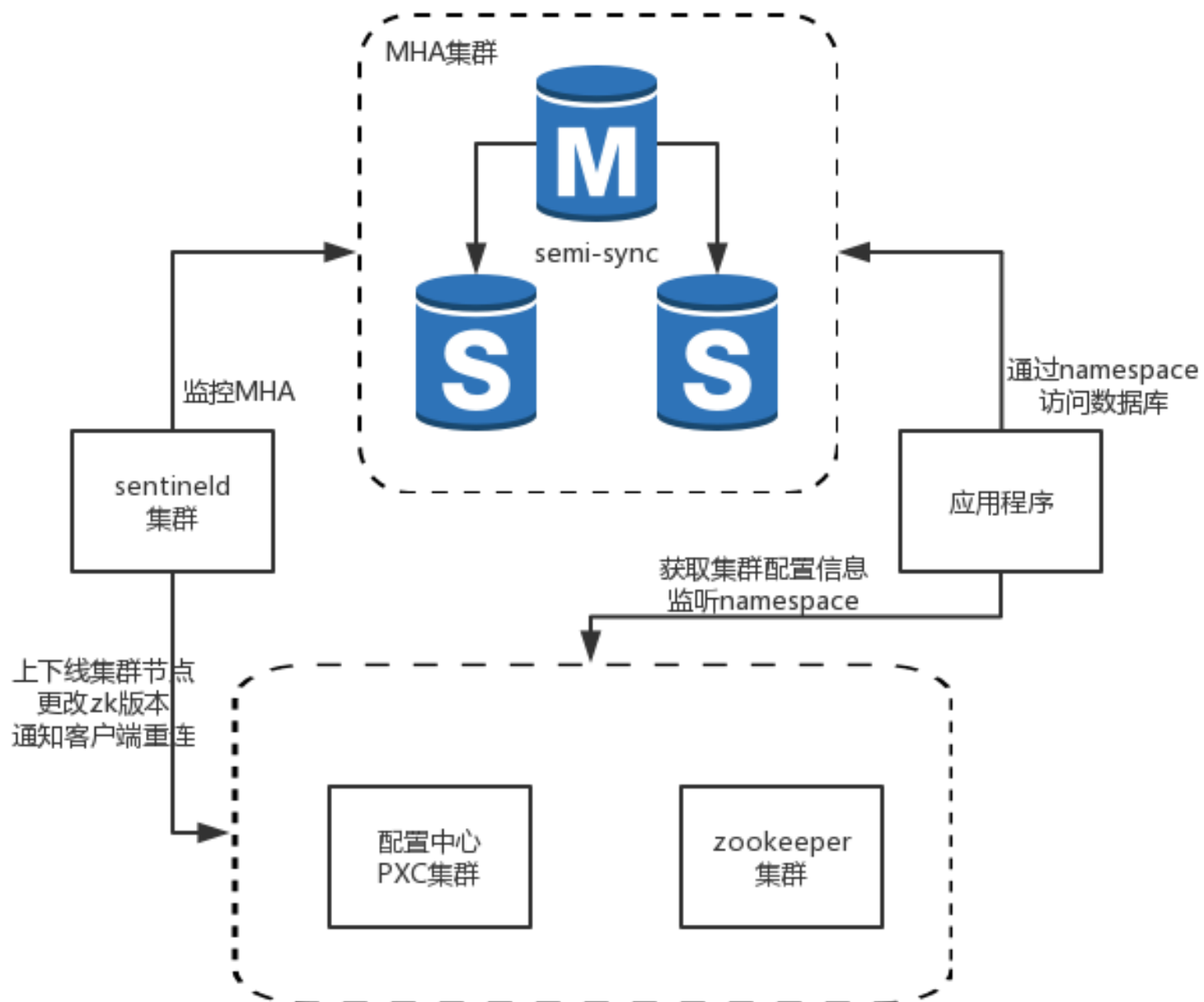
# 3M架构



- monitor节点复制数据库探活和切换
- agent 主要完成monitor发出的切换操作
- 应用程序通过VIP连接

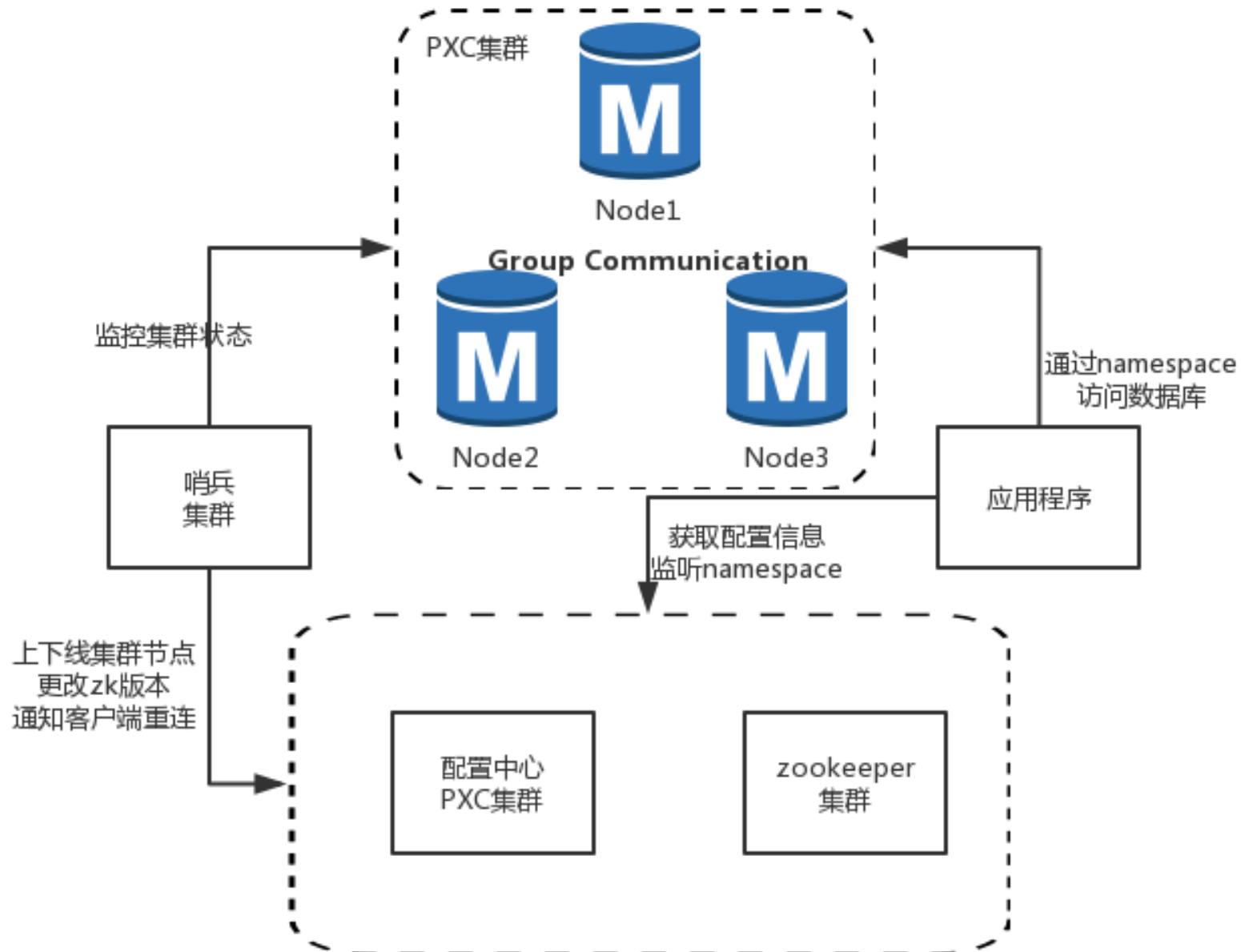
**不再新增!**

# QMHA架构



- 每一个哨兵节点都会监控集群中的所有Mysql节点
- 哨兵节点会相互通信，确认各个MySQL节点的状态
- **连接方式使用namespace 连接**

# PXC架构



- 每一个哨兵节点都会监控集群中的所有Mysql节点
- 哨兵节点会相互通信，确认各个MySQL节点的状态
- **连接方式使用namespace 连接**

## PXC集群使用**注意事项**:

- 不要有大事务更新
- 不要做大查询（如统计查询），若SQL较慢会影响整个集群的性能

# 架构对比

	3M	QMHA	PXC
读写分离	Y	Y	Y
一致性	能保证一致性	半同步复制，能保证数据不丢失	强一致性
自动故障转移	Y	Y	Y
切换影响	会有短暂只读时间	会有短暂只读时间	切换几乎无感知
管理节点	单点	分布式	分布式

**2**

## **MySQL数据库使用规范**

## 二 数据库使用规范

### 1 表的组织结构



### 2 索引组织结构



### 3 事务和锁



### 4 使用规范和注意事项



## 2.1 表的组织结构

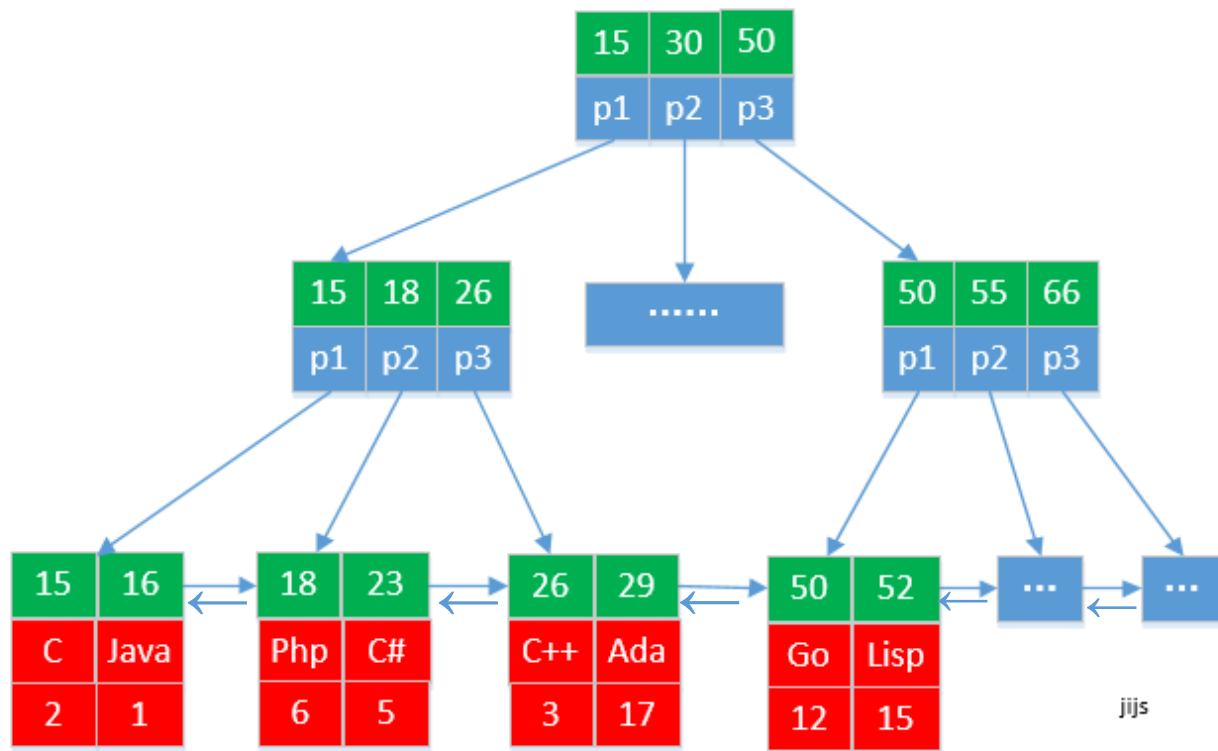
表pl\_ranking包含3个字段，如下：

id: 自增列，主键

plname: 编程语言名称(索引)

ranking: 排名

id	plname	ranking
15	C	2
16	Java	1
18	Php	6
23	C#	5
26	C++	3
29	Ada	17
50	Go	12
52	Lisp	15
...	...	...



表记录都是根据主键顺序组织存放的。

# 索引组织结构

MySQL的索引，一般指二级索引或着辅助索引。

索引的出现其实就是为了提高数据查询的效率，就像书的目录一样。

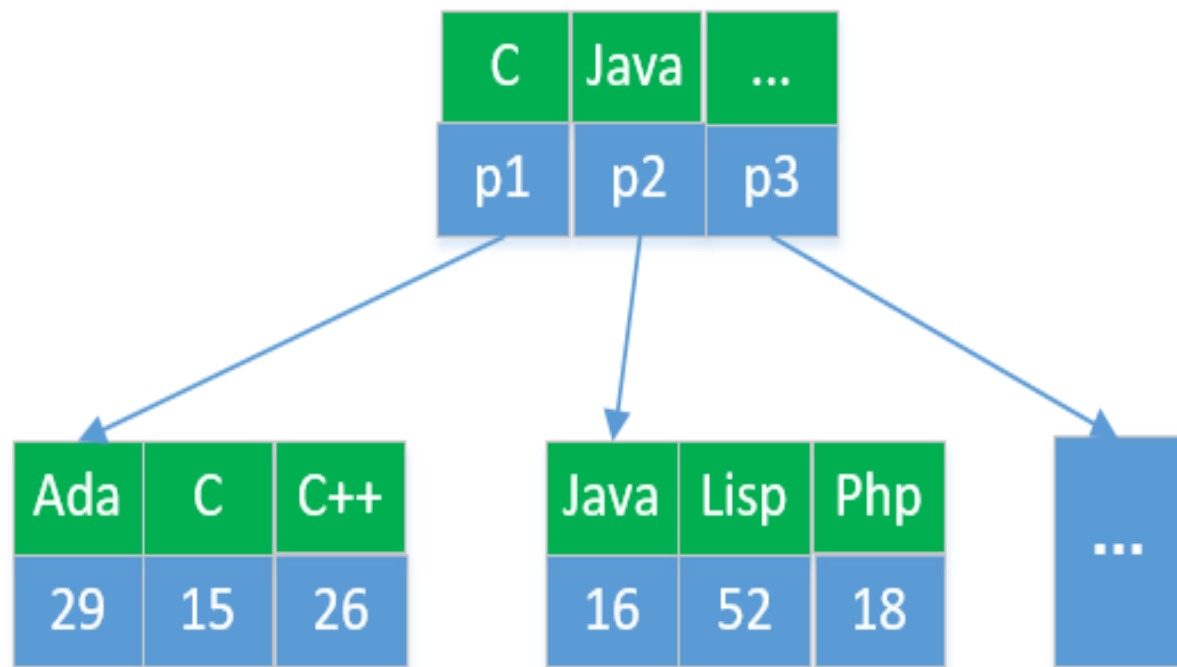
常见的索引类型：

- 普通索引（单列索引）
- 唯一索引
- 前缀索引
- 联合索引
- ...

举例：

在 pl\_ranking 上的pname 上加一个普通索引

```
alter table pl_ranking add key idx_pname(`pname`);
```





## 01

### 普通索引

- 最基本的索引类型，没有任何限制，唯一作用就是加快系统对数据的访问速度
- 适合于经常作为查询条件的列，用作表连接的列
- 创建时注意索引的基数，较低的话一般不适合创建索引

*KEY `idx\_room\_id` (`room\_id`) -- 普通索引*

## 02

### 唯一索引

- 主要为了避免数据出现重复
- 也能加快数据访问速度
- 值必须唯一，但允许有空值

***UNIQUE KEY** `uniq\_telephone` (`telephone`) -- 唯一索引*

## 03

### 前缀索引

- 对键值的前几个（一部分）建立索引
- 适合于字段比较长，而且前几个字符就有很大的区分度
- 不能在 order by 或者 group by 中触发前缀索引，也不能把它们用于覆盖索引

```
KEY `idx_describ` (`room_describe`(10)) --前缀索引
```

## 04

### 联合索引

- 多个列共同组成一个索引
- 适用于某几个特定列经常联合作为查询条件的情况
- 注意索引中列的顺序

```
KEY `idx_create_update` (`create_time`,`update_time`), --联合索引
```

# 索引类型

```
CREATE TABLE `example_index` (  
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT '主键',  
  `room_id` bigint(20) NOT NULL DEFAULT '0' COMMENT 'room_id',  
  `address` varchar(200) NOT NULL DEFAULT '' COMMENT '地址',  
  `room_describe` varchar(200) NOT NULL DEFAULT '' COMMENT '房屋描述',  
  `telephone` bigint(15) NOT NULL DEFAULT '0' COMMENT '电话',  
  `root_features` varchar(200) NOT NULL DEFAULT '' COMMENT '特点',  
  `create_time` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',  
  `update_time` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE  
CURRENT_TIMESTAMP COMMENT '更新时间',  
  PRIMARY KEY (`id`), --主键索引  
  UNIQUE KEY `uniq_telephone` (`telephone`), -- 唯一索引  
  KEY `idx_room_id` (`room_id`), --普通索引  
  KEY `idx_create_update` (`create_time`,`update_time`), --联合索引  
  KEY `idx_describ` (`room_describe`(10)) --前缀索引  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='样例表'
```

# 索引创建原则



## 如何创建好索引呢？

哪些列经常作为where条件？  
哪些列经常用于order by？  
哪些列会和其他表进行关联？  
准备创建索引的列的基数是否较高？

- 经常作为where条件的列建议创建索引
- 和其他表具有关联关系的列建议创建索引
- 能用普通索引不用唯一索引
- 能用单列索引不用联合索引
- 单键索引，选择针对当前query过滤性更好的索引
- 联合索引，where条件中过滤性最好的放在最左侧
- 尽可能选择覆盖索引，避免回表
- 通过分析统计信息和调整query的写法来达到选择合适索引的目的

# 索引失效

## 索引失效的10种场景

不满足最左匹配原则

使用了select \*

索引列上有计算

索引列用了函数

字段类型不同

like左边包含%

列对比

使用or关键字

not in和not exists

order by的坑



# 索引失效

```
mysql> show create table user \G
***** 1. row *****
      Table: user
Create Table: CREATE TABLE `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `code` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
  `age` int(11) DEFAULT '0',
  `name` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
  `height` int(11) DEFAULT '0',
  `address` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `idx_code_age_name` (`code`,`age`,`name`),
  KEY `idx_height` (`height`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4
1 row in set (0.00 sec)

mysql> select * from user ;
+----+-----+-----+-----+-----+-----+
| id | code | age | name  | height | address |
+----+-----+-----+-----+-----+-----+
| 1  | 101  | 21  | 周星驰 | 175    | 香港    |
| 2  | 102  | 18  | 周杰伦 | 173    | 台湾    |
| 3  | 103  | 23  | 苏三   | 174    | 成都    |
+----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

- 不满足最左匹配原则（主要针对联合索引）

```
select * from user
where age=18 and name='周杰伦';
-- 不符合 `idx_code_age_name`
(`code`,`age`,`name`) 的最左匹配原则
```

- select \*

```
select * from user
where name='周杰伦';
```

-- 直接全表扫描

```
select age,name
from user where name='周杰伦';
-- 可以使用idx_code_age_name 索引
```

# 索引失效

```
mysql> show create table user \G
***** 1. row *****
      Table: user
Create Table: CREATE TABLE `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `code` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
  `age` int(11) DEFAULT '0',
  `name` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
  `height` int(11) DEFAULT '0',
  `address` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `idx_code_age_name` (`code`,`age`,`name`),
  KEY `idx_height` (`height`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4
1 row in set (0.00 sec)

mysql> select * from user ;
+----+-----+-----+-----+-----+-----+
| id | code | age | name  | height | address |
+----+-----+-----+-----+-----+-----+
|  1 | 101  |  21 | 周星驰 |    175 | 香港    |
|  2 | 102  |  18 | 周杰伦 |    173 | 台湾    |
|  3 | 103  |  23 | 苏三   |    174 | 成都    |
+----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

- 索引列上有计算  
`select age,name from user  
where height-1 >174;`
- 索引列上使用函数  
`select * from user  
where SUBSTR(height,1,2)=17;`
- 字段类型不同（容易忽略）  
`select * from user  
where code=102;`  
-- code是varchar类型，需要类型转换，索引失效

# 索引失效

```
mysql> show create table user \G
***** 1. row *****
      Table: user
Create Table: CREATE TABLE `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `code` varchar(20) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
  `age` int(11) DEFAULT '0',
  `name` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
  `height` int(11) DEFAULT '0',
  `address` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `idx_code_age_name` (`code`,`age`,`name`),
  KEY `idx_height` (`height`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4
1 row in set (0.00 sec)

mysql> select * from user ;
+----+-----+-----+-----+-----+-----+
| id | code | age | name  | height | address |
+----+-----+-----+-----+-----+-----+
| 1  | 101  | 21  | 周星驰 | 175    | 香港    |
| 2  | 102  | 18  | 周杰伦 | 173    | 台湾    |
| 3  | 103  | 23  | 苏三   | 174    | 成都    |
+----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

- 列对比  
`select * from user  
where age < height ;`
- 使用or  
`select * from user  
where code > '100' or 150 < height ;`
- order by的坑  
`select * from user  
where code = '101'  
order by name, age ;`  
*-- 改成 `order by `age`,`name``*
- like左边使用%  
`select * from user  
where code like '%105%';`



# SQL执行计划

- SQL执行计划

解释该条SQL一步一步怎么执行，在执行中哪些使用索引、哪些进行全表扫描，以及执行顺序等。

通过了解SQL的执行计划可以判断SQL执行效率，对数据库的影响等。

EXPLAIN 适用于 SELECT、DELETE、INSERT、REPLACE 和 UPDATE 语句。

- 如何查看SQL执行计划

explain select age,name from user  
where height >174;

```
id: 1
select_type: SIMPLE
table: user
partitions: NULL
type: range
possible_keys: idx_height
key: idx_height
key_len: 5
ref: NULL
rows: 1
filtered: 100.00
Extra: Using index condition
```

# SQL执行计划

- id: 表示被操作的顺序; id值大, 先被执行, 若id值相同, 执行顺序从上到下
- select\_type: 查询中每个select子句的类型
  - SIMPLE: 简单查询, 不包含 UNION 或者子查询
  - PRIMARY: 查询中如果包含子查询或其他部分, 外层的 SELECT 将被标记为 PRIMARY
  - SUBQUERY: 子查询中的第一个 SELECT
  - UNION: 在 UNION 语句中, UNION 之后出现的 SELECT
  - DERIVED: 在 FROM 中出现的子查询将被标记为 DERIVED
  - UNION RESULT: UNION 查询的结果

# SQL执行计划

- table: 被操作对象的名字, 通常是表名
- partitions: 匹配分区信息 (非分区表为NULL)
- **type: 查询执行的类型, 描述了查询是如何执行的**
  - const: 表中最多只有一行匹配的记录, 常用于使用主键或唯一索引的所有字段作为查询条件
  - eq\_ref: 当连表查询时, 前一张表的行在当前这张表中只有一行与之对应。是除了 system 与 const 之外最好的 join 方式, 常用于使用主键或唯一索引的所有字段作为连表条件
  - ref: 使用普通索引作为查询条件, 查询结果可能找到多个符合条件的行
  - index\_merge: 当查询条件使用了多个索引时, 表示开启了 Index Merge 优化, 此时执行计划中的 key 列列出了使用到的索引
  - range: 对索引列进行范围查询, 执行计划中的 key 列表示哪个索引被使用了
  - index: 全索引扫描, 与 ALL 类似, 只不过扫描的是索引, 而索引一般在内存中, 速度更快。
  - **ALL**: 全表扫描

所有值的顺序从最优到最差排序为: system > const > eq\_ref > ref > fulltext > ref\_or\_null > index\_merge > unique\_subquery > index\_subquery > range > index > ALL

# SQL执行计划

- possible\_keys: 可能被使用的索引
- **key: 真实使用的索引**
- key\_len: 索引键长度, 单位字节
- ref: 表示本行被操作对象的参考对象
- **rows: 扫描的行数 (估计值)**
- **filtered: 符合过滤条件数据的占比**

# SQL执行计划

## ● Extra: 重要的补充信息

- **Using filesort**: 使用文件完成排序
- Using index: 可直接在索引中获取需要的信息。若同时出现Using where表明索引还被用来过滤筛选; 没有出现, 表明只是用来读取数据
- Using index condition: 尝试只使用索引来获取数据, 即能用索引就用
- Using index for group-by: 使用索引优化GROUPBY或者DISTINCT操作, 避免额外的磁盘操作
- **Using temporary**: 用临时表存储中间结果, 常用于 GROUP BY或者 ORDER BY操作
- Using where: 使用了where条件
- Using join buffer (Block Nested Loop): 连表查询的方式, 表示当被驱动表没有使用索引的时候, MySQL 会先将驱动表读出来放到 join buffer 中, 再遍历被驱动表与驱动表进行查询。

这里提醒下, 当 Extra 列包含 Using filesort 或 Using temporary 时, MySQL 的性能可能会存在问题, 需要尽可能避免。

# 事务和锁

事务是指作为单个逻辑工作单元执行的一系列操作，要么完全地执行，要么完全地不执行。

## Atomicity

原子性：指事务是一个不可分割的最小工作单位，事务中的操作只有**都发生和都不发生**两种情况

## Isolation

隔离性：是指事务之间应该是隔离的，并发执行的各个事务之间不能互相干扰，在没有隔离性约束下，并发事务就可能出现脏读、不可重复读、幻读的问题



## Consistency

一致性：一致性指的是数据库业务数据的正确性，事务保证只能把数据库从一个有效（正确）的状态转移到另一个有效（正确）的状态

## Durability

持久性：对于一个已经提交的事务，在事务提交后即使系统发生了崩溃，这个事务对数据库中所做的更改也不能丢失

# 事务和锁

MySQL中的事务控制语句：

## ① 开启事务语句：

- START TRANSACTION
- BEGIN
- SET autocommit=0 (只对当前会话生效)

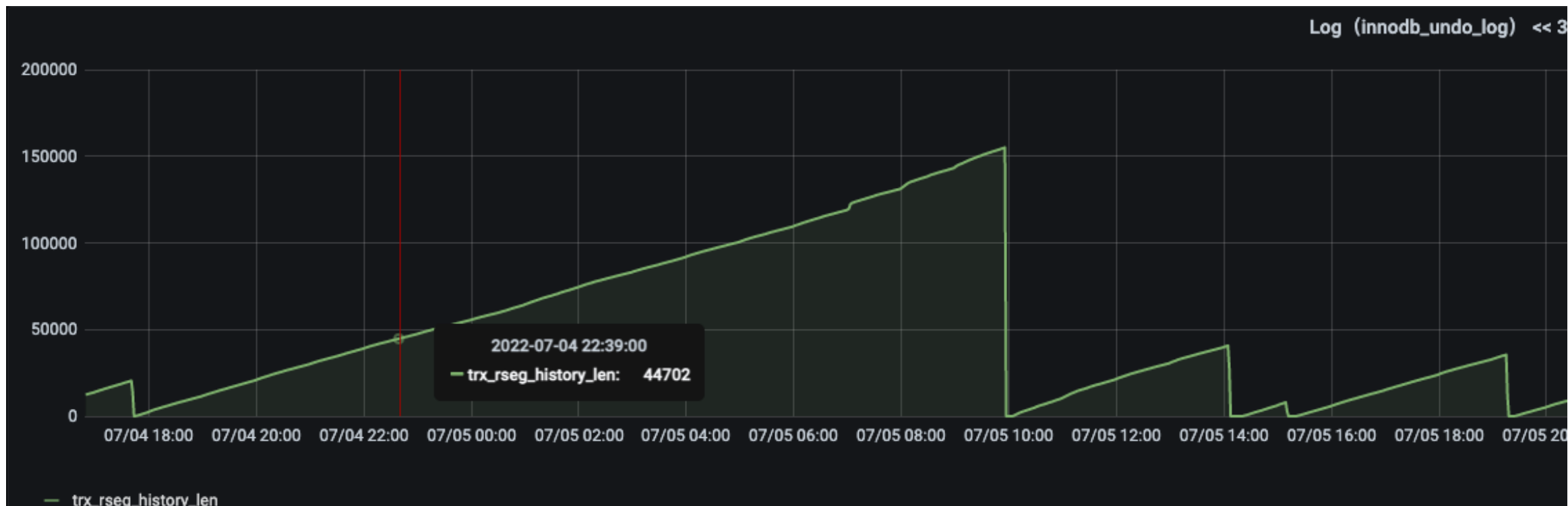
## ② 结束事务语句

- COMMIT：提交当前事务，使更改持久生效
- ROLLBACK：回滚当前事务，取消更改

## 注意：

- 默认情况下，MySQL启用自动提交模式 (autocommit=1)。这意味着，如果没有显示开启事务，则每个语句都是原子的
- 注意在开发过程中使用的SDK中的相关配置，是否开启了自动提交（公司提供的SDK里是默认开启自动提交）
- **显式开启事务，一定要及时关闭。**

# 事务和锁



- 事务长期不关闭，会导致数据库undo log持续增多，进而会导致在其他事务读取该事务涉及的相应数据时**性能极具下降**
- 事务不关闭，其他会话无法对事务相关的表做DDL操作



# 事务和锁

## 什么是锁

- 锁是用来保护数据库中的共享资源（页，行），提供数据的完整性和一致性
- 一般在事务commit和rollback后进行释放

## 锁的类型

MySQL InnoDB中有两种行级锁：

- 共享锁，允许事务读一行数据
- 排他锁，允许事务删除或者更新一行数据

表 6-3 排他锁和共享锁的兼容性

	X	S
X	不兼容	不兼容
S	不兼容	兼容

## 锁的算法

- Record Lock：单个记录上的锁
- Gap Lock：间隙锁，锁定一个范围，但不包含记录本身
- Next-key Lock：Record Lock+Gap Lock，即锁范围，也锁记录本身

## 常见锁的问题

- 锁阻塞
- 死锁

# 使用规范和注意事项

表的创建规范:

- **一定要有主键**，推荐使用bigint unsigned自增列作为主键
- 表字符集使用UTF8MB4字符集
- **不允许单独指定列的字符集**
- 所有字段均定义为NOT NULL
- 尽量不要使用大字段（用TEXT，BLOB等类型）
- 若必须存储大字段、将其拆分到单独的表中存储,分离冷热数据
- 表名和列名统一使用小写，除下划线外不允许有其他特殊字符

索引创建规范:

- **禁止重复索引**
- **禁止冗余索引**
- 不在低基数列上建立索引,例如 “性别”
- 经常在where条件中使用的列，建议加上索引
- 若需创建唯一索引，不建议包含多个列

<https://wiki.corp.qunar.com/confluence/pages/viewpage.action?pageId=43779850>

# 使用规范和注意事项

## SQL使用规范:

- 合理使用覆盖索引减少IO,避免排序
- 用IN代替OR, SQL语句中IN包含的值不应过多
- 禁止使用order by rand()
- order by使用的列, 尽量是where条件中使用的列, 以及索引列
- SELECT只获取必要的字段,禁止使用SELECT \*
- 建议使用合理的分页方式以提高分页效率
- 拆分复杂SQL为多个小SQL,避免大事务
- 对同一个表的多次alter操作必须合并为一次操作
- 避免使用存储过程、触发器、视图、自定义函数等

<https://wiki.corp.qunar.com/confluence/pages/viewpage.action?pageId=43779850>



# 3

## 自动化运维平台使用介绍

# 自动化运维平台使用介绍

## ■ 开发人员主页

- [http://dubai.corp.qunar.com/page#/home\\_dev](http://dubai.corp.qunar.com/page#/home_dev)

## ■ 数据库权限申请

- <http://dubai.corp.qunar.com/page#/databasePermission>

## ■ 数据库资源申请

- <http://dubai.corp.qunar.com/page#/otherApply/createBill>

## ■ SQL审核

- <http://putin.corp.qunar.com/qso/dba/index.html#/home>

## ■ 慢查询查看

- <http://dubai.corp.qunar.com/page#/slowQuery>

## ■ 查询控制台

- <http://dubai.corp.qunar.com/web/console>

# 查询控制台

ip/host/namespace/tujia\_url:

数据库ip/host/namespace/tujia\_url

port (namespace/tujia\_url方式访问无需填写) :

数据库端口

Close

确认

I-collect 2020-08-11 10:33:33

database ▼

information\_schema  
push  
statistics  
crowd\_db  
ata\_intl  
kuaik  
mysql  
performance\_schema  
roduce  
comment  
monitor  
hotice  
push  
report  
sim

1. 填写对应数据库的连接信息
2. 选择对应的 database

说明：查询控制台只提供表的查询功能，无法对表进行修改

# 数据库权限申请

新建工单 ×

在申请权限之前请详细阅读: [帮助文档](#)

新增数据库账号

已有账号扩展IP

已有账号扩展DB

已有账号扩展权限

☒ NAMESPACE ☐ 数据库地址端口

☒ 普通权限申请 ☐ 容器基准权限申请

qmha和pxc使用namespace  
3M使用IP和端口

\* NAMESPACE:

\* 用户名:

\* appcode:

\* 连接范围:

\* DB名称:

\* 权限:

☐ SELECT

☐ SELECT,INSERT,UPDATE,DELETE

☐ CREATE ☐ DROP

☐ REPLICATION SLAVE

☐ REPLICATION CLIENT

提交

- 容器基准权限申请：若应用是使用的容器，则选该选项，无需再填写连接范围
- qmha和pxc架构使用namespace，3M使用和端口
- 用户名：用户名要符合规则
- appcode：应用的appcode，便于记录用户是哪个应用在使用
- 连接范围：应用主机名
- DB名称：要申请访问的database

# 数据库权限申请

新建工单 ×

在申请权限之前请详细阅读: [帮助文档](#)

新增数据库账号

**已有账号扩展IP**

已有账号扩展DB

已有账号扩展权限

☒ NAMESPACE ☐ 数据库地址端口

☒ 普通权限申请 ☐ 容器基准权限申请

\* NAMESPACE:

\* 用户名: 

查看已有权限

\* appcode:

\* 待扩展IP范围:

\* 要扩展的权限编号: 

0

权限编号	用户名称	已有权限主机	数据库	TAB
0	wap_	1-10.10.10.10, 1-10.10.10.10	`qta_`s`	*

提交

- qmha和pxc架构使用namespace, 3M使用和端口
- 用户名: 已有的用户名
- 填写完上面的信息, 可以查询用户现有的权限
- appcode: 应用的appcode, 便于记录用户是哪个应用在使用
- 待扩展IP范围: 新增的应用主机 (IP)
- 扩展权限编号: 对哪一类的用户扩展IP, 会有不同应用主机拥有的权限不一样



# 数据库权限申请

新建工单 ×

在申请权限之前请详细阅读: [帮助文档](#)

新增数据库账号

已有账号扩展IP

已有账号扩展DB

已有账号扩展权限

☒ NAMESPACE ☐ 数据库地址端口

☒ 普通权限申请 ☐ 容器基准权限申请

\* NAMESPACE:

\* 用户名:  查看已有权限

\* appcode:

\* 待扩展DB名称:

\* 要扩展的权限编号:

权限编号	用户名称	已有权限主机	数据库	TABl
0	wap_	<div>1-2, 1-</div>	`qta_`	*

提交

- qmha和pxc架构使用namespace, 3M使用和端口
- 用户名: 已有的用户名
- 填写完上面的信息, 可以查询用户现有的权限
- appcode: 应用的appcode, 便于记录用户是哪个应用在使用
- 待扩展DB名称: 需要权限的database
- 扩展权限编号: 对哪一类的用户扩展IP, 会有不同应用主机拥有的权限不一样

# 数据库权限申请

新建工单

X

在申请权限之前请详细阅读: [帮助文档](#)

新增数据库账号

已有账号扩展IP

已有账号扩展DB

已有账号扩展权限

☒ NAMESPACE ☐ 数据库地址端口

☒ 普通权限申请 ☐ 容器基准权限申请

\* NAMESPACE: qta\_stats

\* 用户名:

wap\_

查看已有权限

\* appcode:

应用appcode

\* 扩展权限:

- ☐ SELECT  
☐ SELECT,INSERT,UPDATE,DELETE  
☐ CREATE ☐ DROP  
☐ REPLICATION SLAVE  
☐ REPLICATION CLIENT

\* 要扩展的权限编号:

▼

权限编号	用户名称	已有权限主机	数据库	TAB
0	wap_	l-a- ph	`qt`	*

提交

- qmha和pxc架构使用namespace, 3M使用和端口
- 用户名: 已有的用户名
- 填写完上面的信息, 可以查询用户现有的权限
- appcode: 应用的appcode, 便于记录用户是哪个应用在使用
- 扩展权限: 需要增加权限
- 扩展权限编号: 对哪一类的用户扩展IP, 会有不同应用主机拥有的权限不一样



# 数据库资源申请

http://dubai.corp.qunar.com/page#/otherApply/createBill

MySQL

Redis

数据库申请

PxcBeta环境新增

QmhaBeta环境新增

MySQL归档申请

DBA操作申请

MySQL数据恢复申请

MySQL离线库申请

在申请数据库之前请详细阅读：[帮助文档](#)

工单标题：

工单名称

数据库名称：

需要创建的database

应用appcode：

数据库所属部门：

机票

应用读比例：

1

应用写比例：

1

应用的读写比例情况，是读多写少，还是写多读少，做一个比例测算

qps：

应用高峰期达到的预估QPS

应用类型：

OLTP

应用说明：

对应用类型，功能，重要性做一个说明

宕机影响：

数据库宕机对业务和应用的影响

宕机故障级别：

P1

宕机后对应用和业务的影响重要程度，P1最高

是否内部系统：

是否为公司内部使用的系统

部署机房地址：

cn0

各表详细信息：

表名	年增长量	表最大数据量	表列总量
预估一下表的最大体量，增长速度，以便DBA更好地选择资源			
<div><div></div><div>暂无数据</div></div>			

增加表信息

清空表信息

提交

# 数据库资源申请

http://dubai.corp.qunar.com/page#/otherApply/createBill

MySQL

Redis

数据库申请

PxcBeta环境新增

QmhaBeta环境新增

MySQL归档申请

DBA操作申请

MySQL数据恢复申请

MySQL离线库申请

在申请pxc测试环境的NAMESPACE之前请详细阅读：[帮助文档](#)

工单标题：

数据库所属部门：

机票

beta-namespace：

NAMESPACE	IP	PORT
填写对应的namespace，beta环境的数据库的IP和port		
<div><div></div><div>暂无数据</div></div>		

增加表信息

清空表信息

提交

# 数据库资源申请

http://dubai.corp.qunar.com/page#/otherApply/createBill

MySQL

Redis

数据库申请

PxcBeta环境新增

QmhaBeta环境新增

MySQL归档申请

DBA操作申请

MySQL数据恢复申请

MySQL离线库申请

数据库归档申请前请详细阅读：[帮助文档](#)

工单标题：

工单名称

数据库地址：

namespace

需要归档数据的实例namespce

数据库名称：

database

需要归档表的database

业务部门：

支付

归档表信息：

表名	归档表名	归档条件	归档类型
tab1		update_time<'2020-07-01 00:00:00'	普通归档

增加表信息

清空表信息

提交

数据归档就是把线上的一些历史数据，归档到历史库，以达到线上数据量的缩容

归档类型：

- 1.普通归档
- 2.定时归档
3. 仅迁移不删除源端数据
- 4.全表归档删除源表
- 5.全表归档保留表结构

# SQL审核

<http://putin.corp.qunar.com/qso/dba/index.html#/home>

The screenshot shows a web interface for SQL review. At the top, there is a 'NAMESPACE' field with the value 'asp\_pms'. Below it, there are radio buttons for '提交方式' (Submit Method) with '文本' (Text) selected. The '执行语句' (Execute Statement) section contains a text area with SQL code: 'set names utf8;', 'use xxx;', and 'CREATE | ALTER | INSERT | UPDATE | DELETE'. At the bottom, there is a '辅助审核语句 (待执行)' (Auxiliary Review Statement (Pending Execution)) section with a '选择文件' (Select File) button and the text '未选择任何文件'. A '检测' (Check) button is located at the bottom right.

只需要填写namespace  
不需要管IP或者端口了

提交方式 ☒ 文本 ☐ 附件

执行语句

本次申请需要执行的语句

set names utf8;  
use xxx;  
CREATE | ALTER | INSERT | UPDATE | DELETE

一般第一行  
运行sql编码  
set names utf8mb4 或者 utf8  
第二行 use dbname  
第三行起 执行的sql

辅助审核语句 (待执行)

选择文件 未选择任何文件

辅助的，新增表，改表，索引等提供，对应的增删改查sql，用于dba判断是否合理，文件必须是utf8格式

辅助审核语句附件请上传跟执行语句里面涉及的表相关的带where条件的SQL，  
以方便DBA判断是否添加索引！  
支持格式：.sql .txt .xml

检测

- 同一个实例（namespace）的多条SQL合并到一个工单
- 同一张表的多个DDL操作，合并成同一条DDL语句

线上环境使用的DDL工具是pt-osc，相关介绍可查看该文章：  
[https://mp.weixin.qq.com/s/gDxmSjWBO6Zmy\\_R5vBwLQw](https://mp.weixin.qq.com/s/gDxmSjWBO6Zmy_R5vBwLQw)

# 数据库权限申请

新建工单

X

在申请权限之前请详细阅读: [帮助文档](#)

新增数据库账号

已有账号扩展IP

已有账号扩展DB

已有账号扩展权限

☒ NAMESPACE ☐ 数据库地址端口

☒ 普通权限申请 ☐ 容器基准权限申请

\* NAMESPACE:

\* 用户名:  [查看已有权限](#)

\* appcode:

\* 扩展权限:

- ☐ SELECT
- ☐ SELECT,INSERT,UPDATE,DELETE
- ☐ CREATE ☐ DROP
- ☐ REPLICATION SLAVE
- ☐ REPLICATION CLIENT

\* 要扩展的权限编号:

权限编号	用户名称	已有权限主机	数据库	TABl
0	wap_...	l-a... l- ph...	`qt...`	*

提交

- qmha和pxc架构使用namespace, 3M使用和端口
- 用户名: 已有的用户名
- 填写完上面的信息, 可以查询用户现有的权限
- appcode: 应用的appcode, 便于记录用户是哪个应用在使用
- 扩展权限: 需要增加权限
- 扩展权限编号: 对哪一类的用户扩展IP, 会有不同应用主机拥有的权限不一样



# 课程总结

- 熟悉公司内部MySQL数据库架构，工作中注意不同架构对应不同的客户端连接
- 了解MySQL表和索引结构，工作中结合使用规范进行表和索引设计
- 熟悉索引失效的情况，工作中一定要避免
- 了解事务特性和使用时注意事项
- 了解锁的概念，工作中注意锁的争用情况
- 熟悉数据库自动化运维平台

**敬畏生产环境！**

**所有的线上操作一定要仔细检查和多次确认！**





# Thanks

To be coding, to be working!