咕泡学院
gupaoedu.com

VIP

【VIP直播课】

MySQL事务与锁详解

青山

专业IT教育培训，做技术人的指路明灯，职场生涯的精神导师
咕泡学院官网:http://www.gupaoedu.com

关注咕泡
码上升职加薪

青山

曾就职于国内顶级FinTech公司

多年Java从业经验，常年从事金融领域的项目研发，拥有传统金融和大型互联网金融项目的架构设计经验。经历了消费金融公司从单体架构到大型分布式、微服务系统的演变，支撑千万级客户和日均过亿放款量。

课程目标

1、掌握事务的特性与事务并发造成的问题
2、事务读一致性问题的解决方案
3、MVCC的原理
4、锁的分类、行锁的原理、行锁的算法

适合了解MySQL和InnoDB架构、掌握索引本质的同学学习。

1

什么是数据库的事务？

数据库事务的典型场景？
什么是事务？
哪些存储引擎支持事务？
事务的四大特性？
数据库什么时候会出现事务？
事务并发会带来什么问题？

事务是数据库管理系统（DBMS）执行过程中的一个**逻辑单位**，由一个有限的数据库**操作序列**构成。

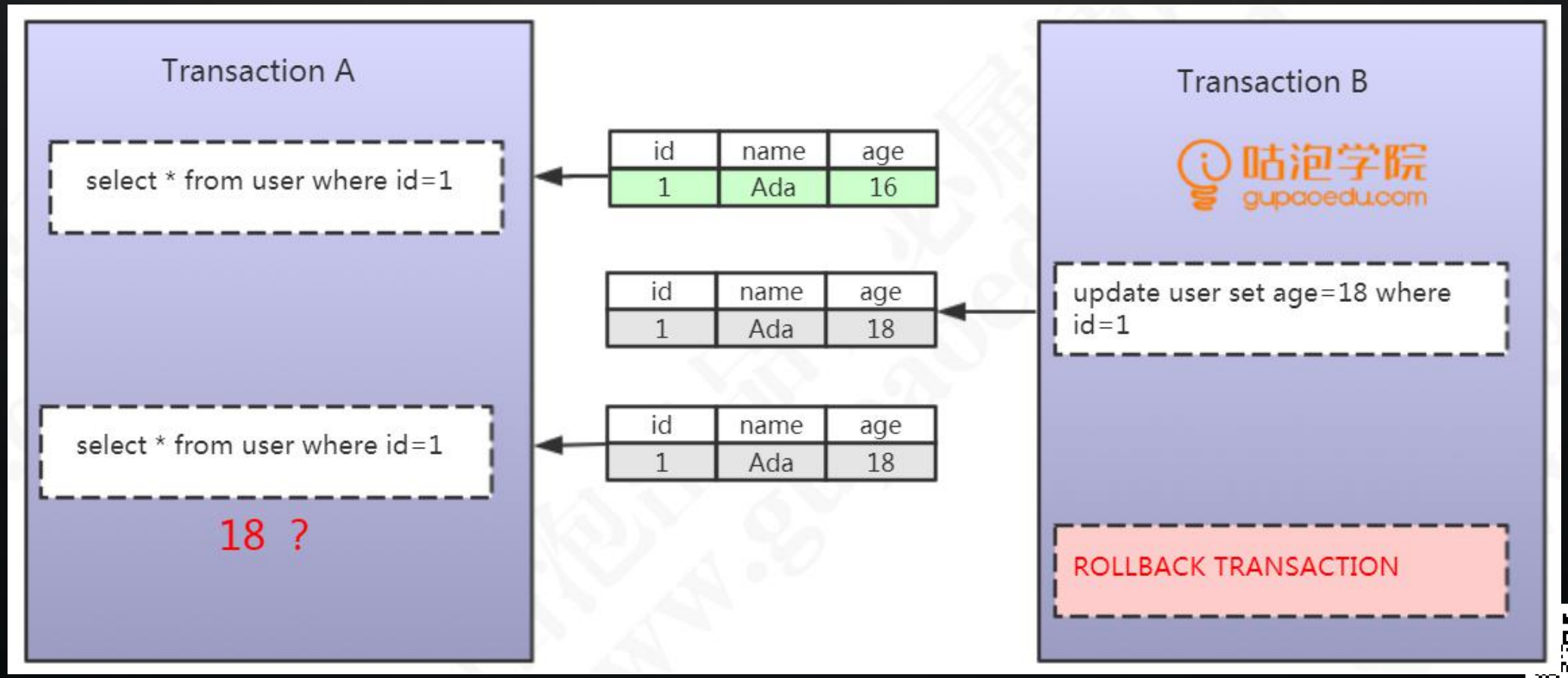
- 原子性（Atomicity） [ˌætəˈmɪsɪti] undo log
- 一致性（Consistent） [kənˈsɪstənt]
- 隔离性（Isolation） [ˌaɪsəˈleɪʃn]
- 持久性（Durable） [ˈdjʊərəbl]

MySQL中如何开启事务：

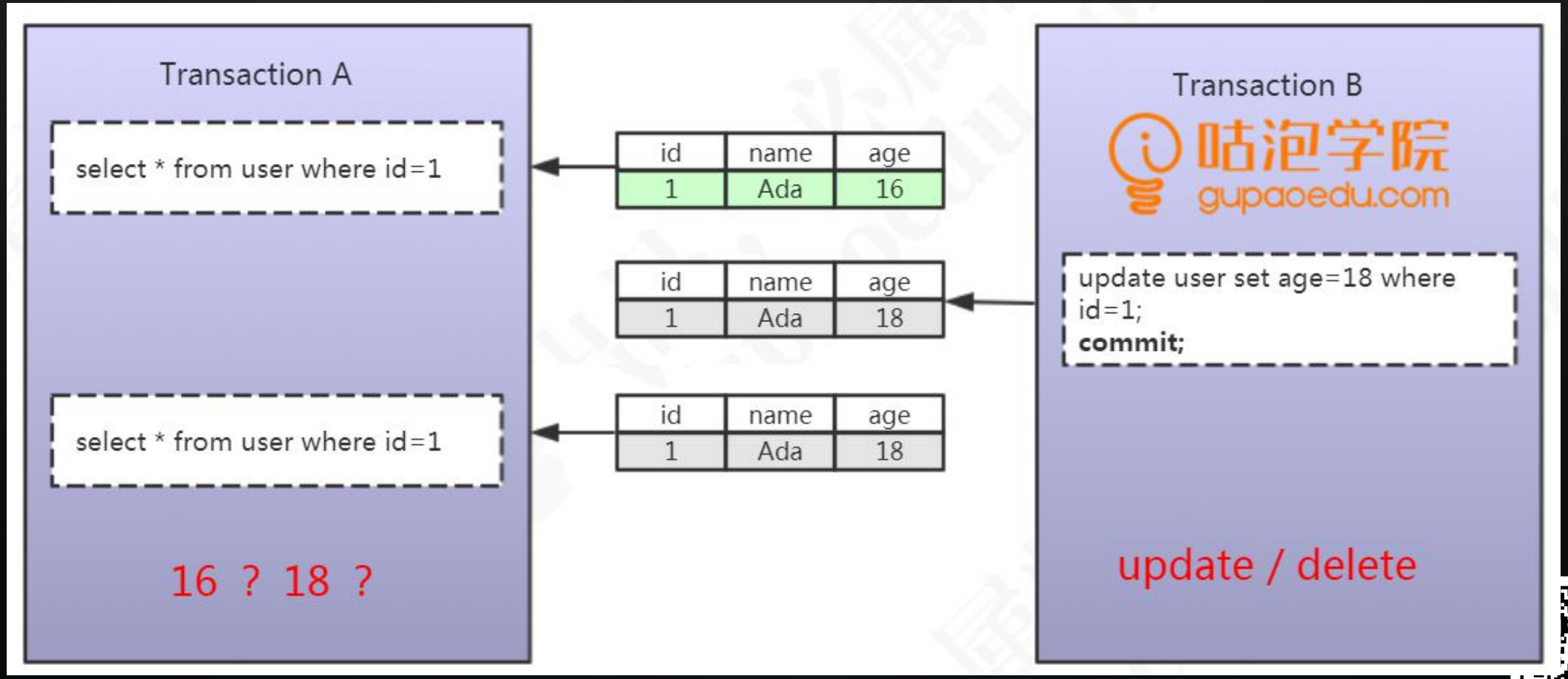
```
set session autocommit = on/off;    -- 设定事务是否自动开启
begin / start transaction           -- 手工方式

commit / rollback                   -- 事务提交或回滚
```

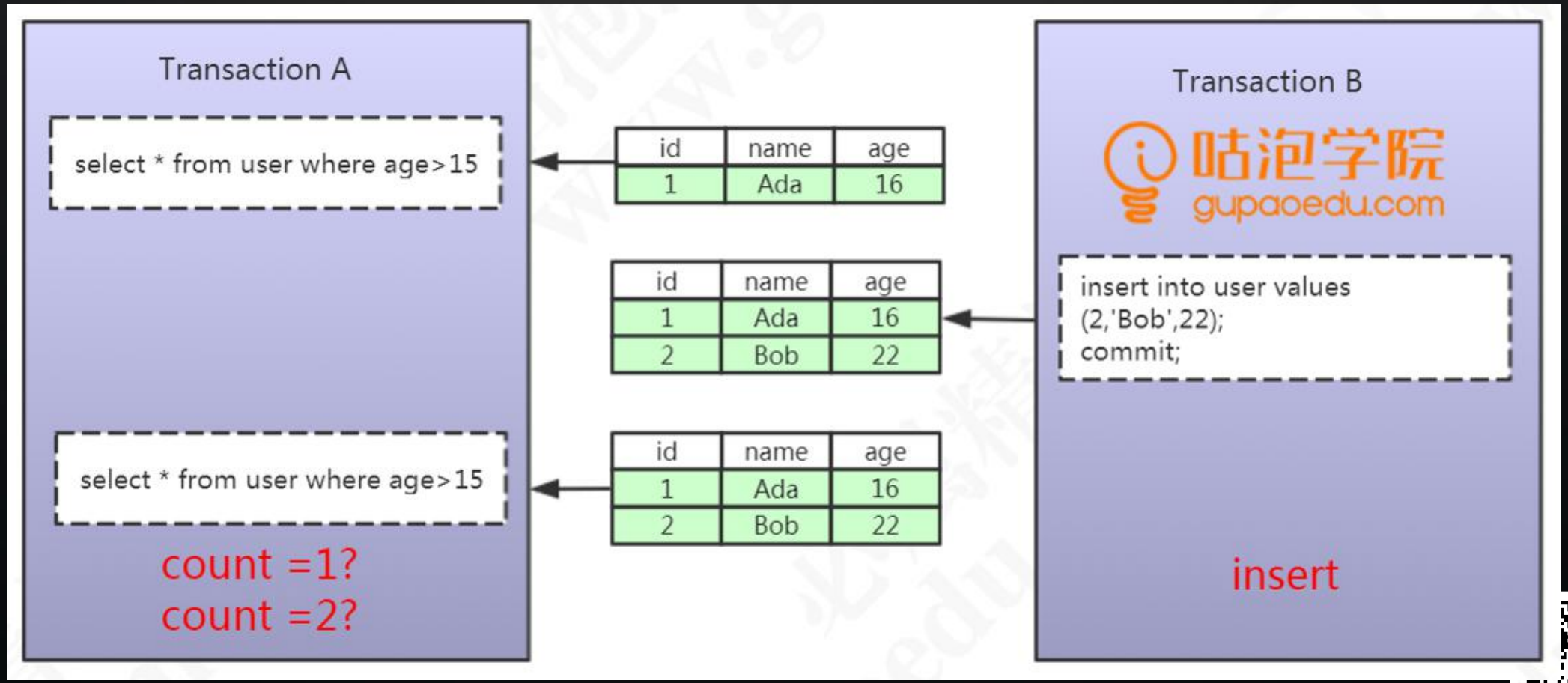
脏读



不可重复读



幻读



事务并发的三大问题其实都是数据库读一致性问题，必须由数据库提供一定的事务隔离机制来解决。

事务的四种隔离级别

SQL92 ANSI/ISO标准：
http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt

Read Uncommitted（未提交读）　　--未解决任何并发问题
事务未提交的数据对其他事务也是可见的，会出现脏读

Read Committed（已提交读）　　--解决脏读问题
一个事务开始之后，只能看到已提交的事务所做的修改，会出现不可重复读

Repeatable Read (可重复读)　　--解决不可重复读问题
在同一个事务中多次读取同样的数据结果是一样的，这种隔离级别未定义解决幻读的问题

Serializable（串行化）　　--解决所有问题
最高的隔离级别，通过强制事务的串行执行

MySQL InnoDB对事务隔离级别的支持程度

隔离级别
并发度

事务隔离级别	脏读	不可重复读	幻读
未提交读（Read Uncommitted）	可能	可能	可能
已提交读（Read Committed）	不可能	可能	可能
可重复读（Repeatable Read）	不可能	不可能	对 InnoDB 不可能
串行化（Serializable）	不可能	不可能	不可能

如果要解决读一致性的问题，
保证一个事务中前后两次读取数据结果一致，实现事务隔离，
应该怎么做？

解决方案：

第一种：在读取数据前，对其加锁，阻止其他事务对数据进行修改（LBCC） Lock Based Concurrency Control。

第二种：生成一个数据请求时间点的一致性数据快照（Snapshot)，并用这个快照来提供一定级别（语句级或事务级）的一致性读取（MVCC）Multi Version Concurrency Control。

DB_TRX_ID，6字节：插入或更新行的最后一个事务的事务ID，事务编号是自动递增的（创建版本）。

DB_ROLL_PTR，7字节：回滚指针（删除版本）。

id	name	DB_TRX_ID	DB_ROLL_PTR
1	aaa	01	NULL

咕泡学院
gupaoedu.com

2

MySQL InnoDB 锁的基本类型

专业IT教育培训，做技术人的指路明灯，职场生涯的精神导师
咕泡学院官网:http://www.gupaoedu.com

关注咕泡
码上升职加薪

表锁与行锁的区别：

锁定粒度：表锁 > 行锁

加锁效率：表锁 > 行锁

冲突概率：表锁 > 行锁

并发性能：表锁 < 行锁

问题：MyISAM和InnoDB分别支持什么粒度的锁？

共享锁（行锁）：Shared Locks

排它锁（行锁）：Exclusive Locks

意向共享锁（表锁）：Intention Shared Locks

意向排它锁（表锁）：Intention Exclusive Locks

行锁算法

- 记录锁 Record Locks
- 间隙锁 Gap Locks
- 临键锁 Next-key Locks

行锁：共享锁(Shared Locks)

共享锁:

又称为读锁，简称S锁，顾名思义，共享锁就是多个事务对于同一数据可以共享一把锁，都能访问到数据，但是只能读不能修改;

加锁释锁方式：

```
select * from student where id=1 LOCK IN SHARE MODE;

commit/rollback;
```


排他锁:

又称为写锁，简称X锁，排他锁不能与其他锁并存，如一个事务获取了一个数据行的排他锁，其他事务就不能再获取该行的锁（共享锁、排他锁），只有该获取了排他锁的事务是可以对数据行进行读取和修改。

加锁释锁方式：

自动：delete / update / insert 默认加上X锁；

手动：select * from student where id=1 FOR UPDATE;

commit/rollback;

意向锁是由数据引擎自己维护的，用户无法手动操作意向锁 。

意向共享锁（Intention Shared Lock，简称IS锁）

表示事务准备给数据行加入共享锁，也就是说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（Intention Exclusive Lock，简称IX锁）

表示事务准备给数据行加入排他锁，说明事务在一个数据行加排他锁前必须先取得该表的IX锁。

思考：为什么需要（表级别的）意向锁？

锁的作用：？

锁到底锁住了什么？

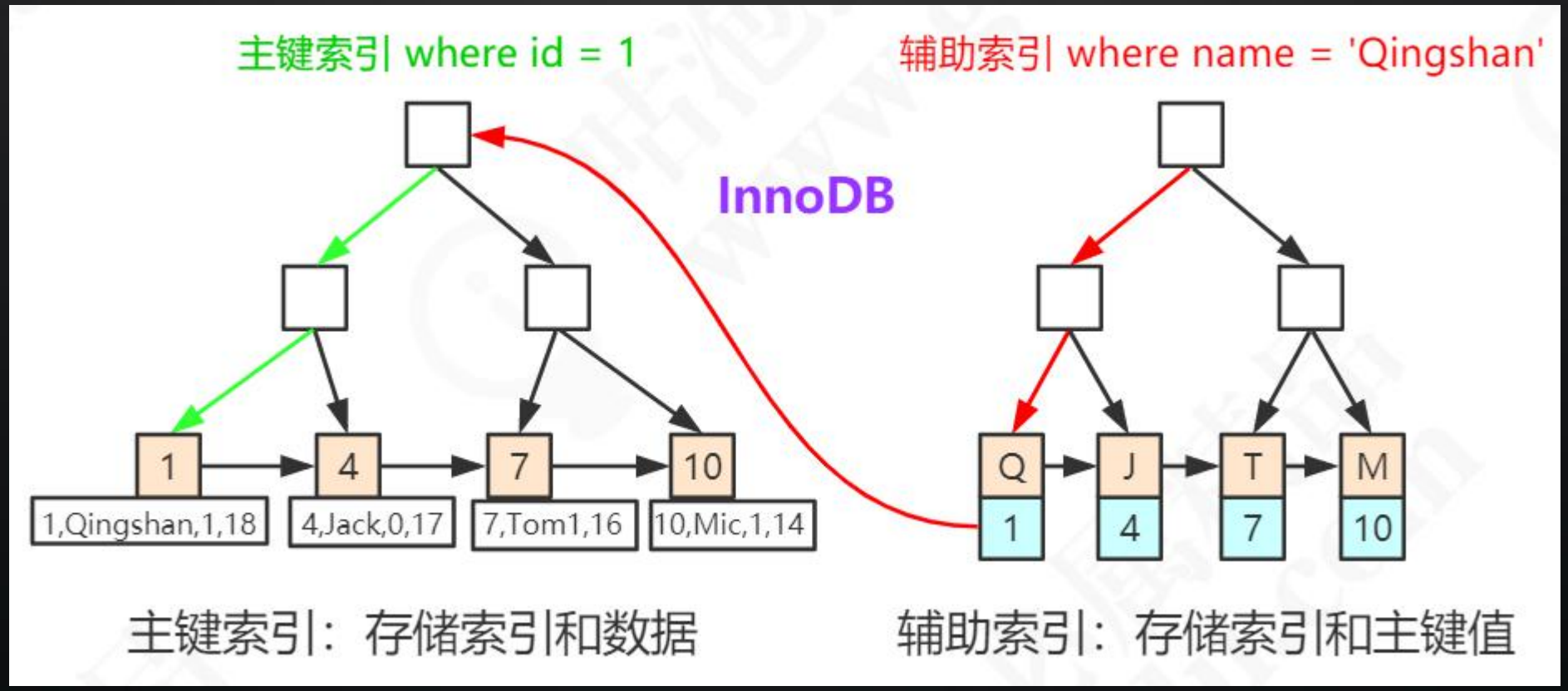
- 是一行数据（Row ）吗？
- 是一个字段（Column）吗？

3

锁的原理：到底锁住了什么？

1、不使用索引　　t1
2、主键索引　　t2
3、唯一索引　　t3

主键索引和辅助索引（InnoDB）



4

行锁的算法：锁住了什么范围？

带主键建表语句

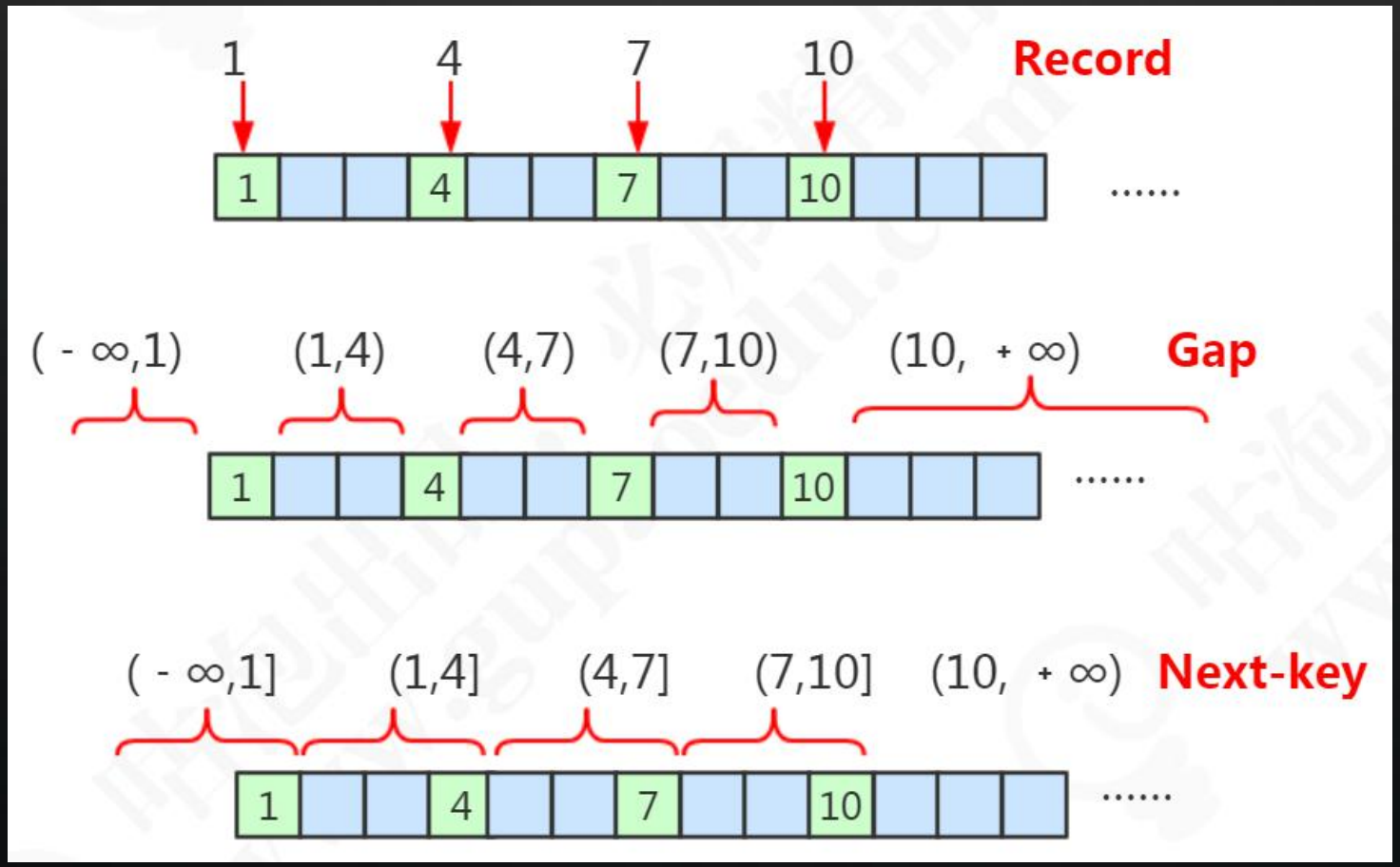

```
DROP TABLE IF EXISTS `t2`;
CREATE TABLE `t2` (
  `id` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
INSERT INTO `t2` (`id`, `name`) VALUES (1, '1');
INSERT INTO `t2` (`id`, `name`) VALUES (4, '4');
INSERT INTO `t2` (`id`, `name`) VALUES (7, '7');
INSERT INTO `t2` (`id`, `name`) VALUES (10, '10');
```


记录

间隙

临键

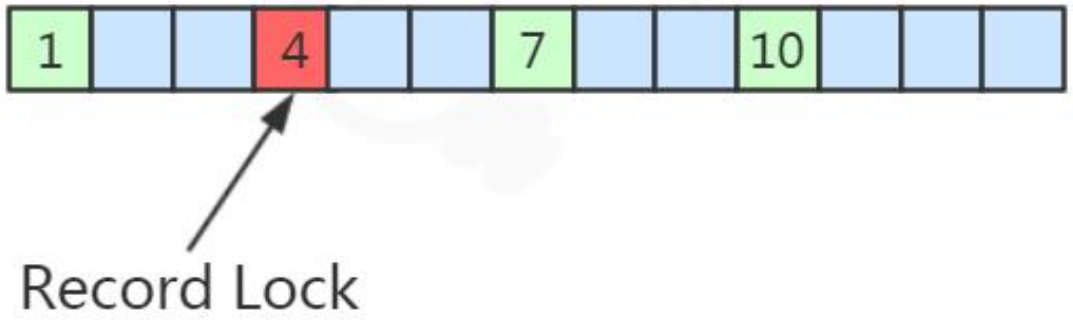


思考：字符可以排序吗？

记录锁(Record Lock)：锁定记录

Record Lock：记录锁

唯一性索引（唯一/主键）等值查询，精准匹配



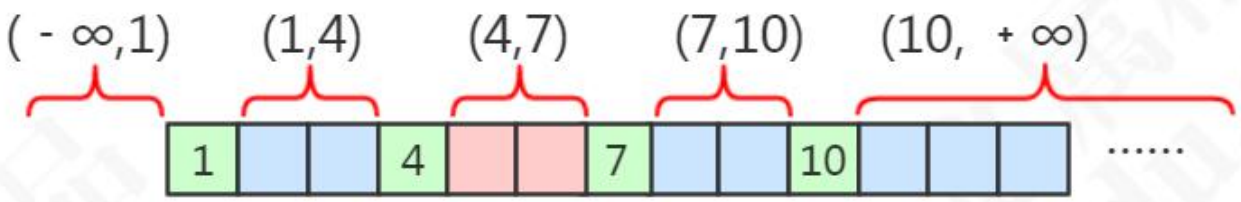

```
select * from t2
where id =4 for update;
```

锁住：
id=4

间隙锁(Gap Lock)：锁定范围

Gap Lock：间隙锁

条件：记录不存在



注意：Gap锁之间不冲突

select * from t2 where id >4 and id <7 for update;

select * from t where id =6 for update;

锁住：
(4,7)

select * from t2 where id >20 for update;

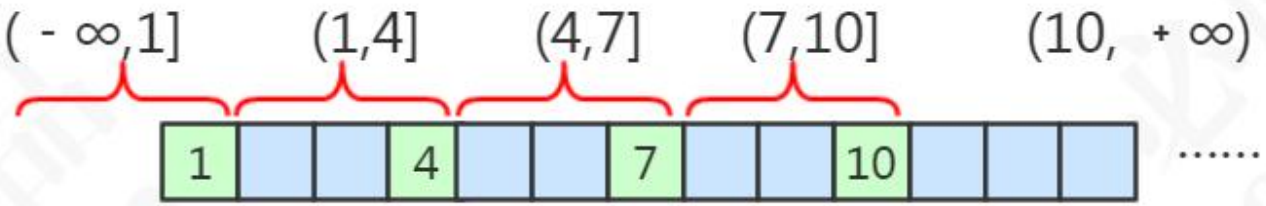
锁住：
(10,+∞)

临键锁(Next-key Lock)：锁定范围加记录

Next-Key Lock：临键锁

条件：范围查询，包含记录和区间

```
select * from t2 where
id >5 and id <9 for update;
```



锁住：
(4,7]　(7,10]

Next-key Lock = Gap Lock + Record lock

InnoDB事务隔离级别的实现

事务隔离级别	脏读	不可重复读	幻读
未提交读（Read Uncommitted）	可能	可能	可能
已提交读（Read Committed）	不可能	可能	可能
可重复读（Repeatable Read）	不可能	不可能	对 InnoDB 不可能
串行化（Serializable）	不可能	不可能	不可能

Read Uncommited

不加锁

Serializable

所有的select语句都会被隐式的转化为select ... in share mode，会和update、delete互斥。

事务隔离级别的实现

	RR	RC
普通的select	MVCC	MVCC
加锁的select和更新 select ... in share mode select ... for update insert update delete	Record Lock Gap Lock Next-key Lock	Record Lock *

事务隔离级别如何选择？

1、RR的间隙锁会导致锁定范围的扩大。

2、条件列未使用到索引，RR锁表，RC锁行。

3、RC的“半一致性”（semi-consistent）读可以增加update操作的并发性。

死锁的发生

1）互斥

2）不可剥夺

3）形成等待环路

死锁的避免

1）顺序访问

2）数据排序

3）申请足够级别的锁

4）避免没有where条件的操作

5）大事务分解成小事务

6）使用等值查询而不是范围查询

咕泡学院
gupaoedu.com

谢谢你的鼓励和支持

青山老师

作业