

# MVCC

# 目录

1 MVCC基本概念

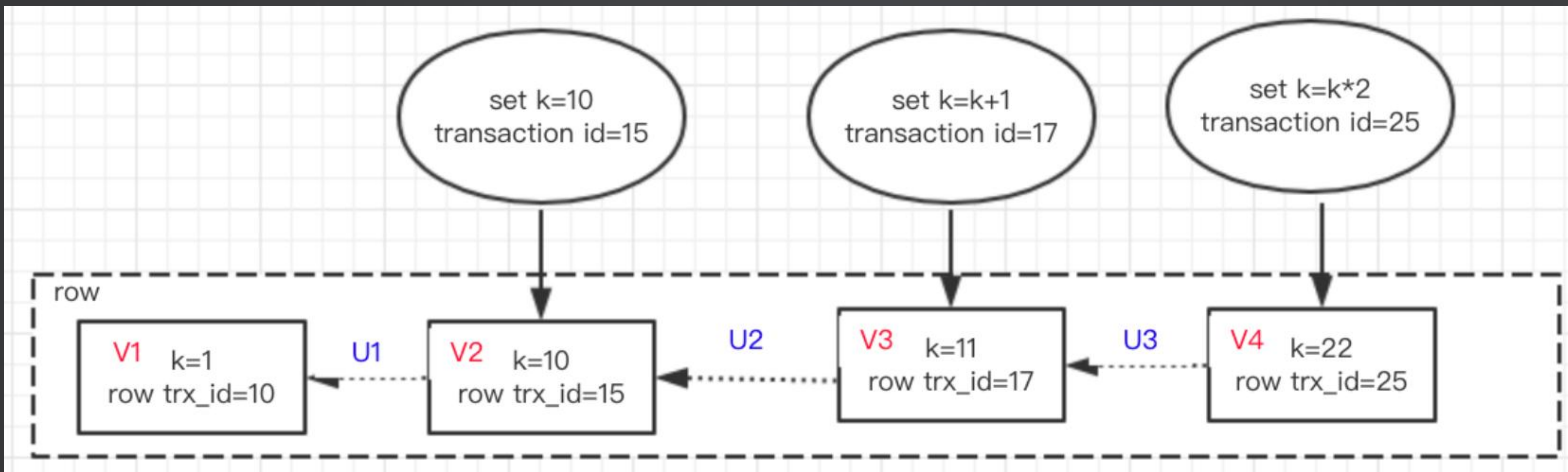
2 实践和使用场景

# 基础知识回顾

- --transaction-isolation=**level**

Command-Line Format	--transaction-isolation=name
System Variable	transaction_isolation
Scope	Global, Session
Dynamic	Yes
<u>SET VAR</u> Hint Applies	No
Type	Enumeration
Default Value	REPEATABLE-READ
Valid Values	READ-UNCOMMITTED READ-COMMITTED REPEATABLE-READ SERIALIZABLE

# 多版本的物理形态



# 分段

时刻	id=1	begin	begin	begin	begin
T0					
T1	set k=1				
T2		selet id=1			
T3	set k=10				
T4			selet id=1		
T5	set k=k+1				
T6				selet id=1	
T7	set k=k*2				
T8		selet id=1 //k= <b>1</b>	selet id=1 //k= <b>10</b>	selet id=1 //k= <b>11</b>	selet id=1 //k= <b>22</b>



# 基本例子

Session1 (RR)

```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from t where id=3;
+----+----+-----+-----+
| id | a | b      | c      |
+----+----+-----+-----+
| 3  | 3 |      3 |      3 |
+----+----+-----+-----+
1 row in set (0.00 sec)
```

Session 2(RR)

```
mysql> update t set c=c+1 where id=3;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from t where id=3;
+----+----+-----+-----+
| id | a | b      | c      |
+----+----+-----+-----+
| 3  | 3 |      3 |      3 |
+----+----+-----+-----+
1 row in set (0.00 sec)

mysql> update t set c=c*10 where id=3 and c=3;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0
```

# 读提交

Session1 (RC)

```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from t where id=3;
+-----+-----+-----+-----+
| id | a | b | c |
+-----+-----+-----+-----+
| 3 | 3 | 3 | 3 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Session 2

```
mysql> update t set c=c+1 where id=3;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from t where id=3;
+-----+-----+-----+-----+
| id | a | b | c |
+-----+-----+-----+-----+
| 3 | 3 | 3 | 4 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```



# 课堂练习：理解RC && RR

```
#cat bash.sh
mysql ... test -e "set transaction_isolation='READ-COMMITTED';begin;select id,c,sleep(1) from t union all select
id,c,0 from t;" &
sleep 0.5
mysql ... test -e "update t set c=c*10 where id=3"
```

id	c	sleep(1)
1	1	0
2	2	0
3	3	0
4	4	0
id	c	
1	1	
2	2	
3	30	
4	4	

session 1 (RC)	
begin select .. sleep(1).. read row1 start	
	sleep 0.5 update c=c*10 where id=3
read row1 end read row2/row3/row4	
select id,c from t; read row1 start read row1 end read row2/row3/row4	



# 课堂练习：理解RC && RR

```
#cat bash.sh
mysql ... test -e "set transaction_isolation='REPEATABLE-READ;begin;select id,c,sleep(1) from t;select id,c from t;" &
sleep 0.5
mysql ... test -e "update t set c=c*10 where id=3"
```

id	c	sleep(1)
1	1	0
2	2	0
3	3	0
4	4	0

id	c
1	1
2	2
3	3
4	4

session 1 (RR)	
begin; select .. sleep(1).. read row1 start	
	sleep 0.5 update c=c*10
read row1 end read row2/row3/row4	
select id,c from t; read row1 start read row1 end read row2/row3/row4	



# 课堂练习

Session1 (RR)	Session 2(RR)
<pre>begin; select * from t where id&gt;=1 limit 2; id a b c 1  1 1 1 2  2 2 2 //0.00s</pre>	
	<pre>for i in (1, 1000000):     update t set c=c+1 where id=2</pre>
<pre>select * from t where id&gt;=1 limit 2; id a b c</pre> <div>显示什么?</div> <pre>select * from t where id&gt;=1 limit 1; id a b c 1  1 1 1</pre>	<div> A) <pre>id a b c 1  1 1 1 2  2 2 2 //0.00s</pre> </div> <div> B) <pre>id a b c 1  1 1 1 2  2 2 2 //1.00s</pre> </div> <div> C) <pre>id a b c 1  1 1 1 2  2 2 1000002 //0.00s</pre> </div>





# 分段

Session1 (RR)

```
begin;
select * from t where id>=1 limit 2;
id a b c
1  1 1 1
2  2 2 2
//0.00s
```

Session 2(RR)

```
for i in (1, 1000000):
    update t set c=c+1 where id=2
```

```
select * from t where id>=1 limit 2;
id a b c
```

显示什么?

```
select * from t where id>=1 limit 1;
id a b c
1  1 1 1
```

A)

```
id a b c
1  1 1 1
2  2 2 2
//0.00s
```

B)

```
id a b c
1  1 1 1
2  2 2 2
//1.00s
```

C)

```
id a b c
1  1 1 1
2  2 2 1000002
//0.00s
```



# 备份

假设业务库里有2个myisam表，其他的都是InnoDB表，

my1(myisam)

my2(myisam)

table balance (innodb)

table records (innodb)

需求：逻辑备份这个库

## 方案1

start transaction with consistent snapshot;

select my1

select my2

select balance

select records

commit;

## 方案2

flush tables with read lock;

select my1

select my2

select balance

select records

unlock tables;

# 备份

假设业务库里有2个myisam表，其他的都是InnoDB表，

my1(myisam)

my2(myisam)

table balance (innodb)

table records (innodb)

需求：逻辑备份这个库

## 方案1

start transaction with consistent snapshot;

select my1

select my2

select balance

select records

commit;

## 方案2

flush tables with read lock;

select my1

select my2

select balance

select records

unlock tables;



# 备份

假设业务库里有2个myisam表，其他的都是InnoDB表，

my1(myisam)

my2(myisam)

table balance (innodb)

table records (innodb)

需求：逻辑备份这个库

## 方案3

**flush tables with read lock;**

select my1

select my2

**start transaction with consistent snapshot;**

**unlock tables;**

select balance

select records

**commit;**

## 方案4 for htap or others

**flush tables with read lock;**

//snapshot for engine1

//snapshot for engine2

**start transaction with consistent snapshot;**

**unlock tables;**

select balance

select records

select tables of engineX

**commit;**

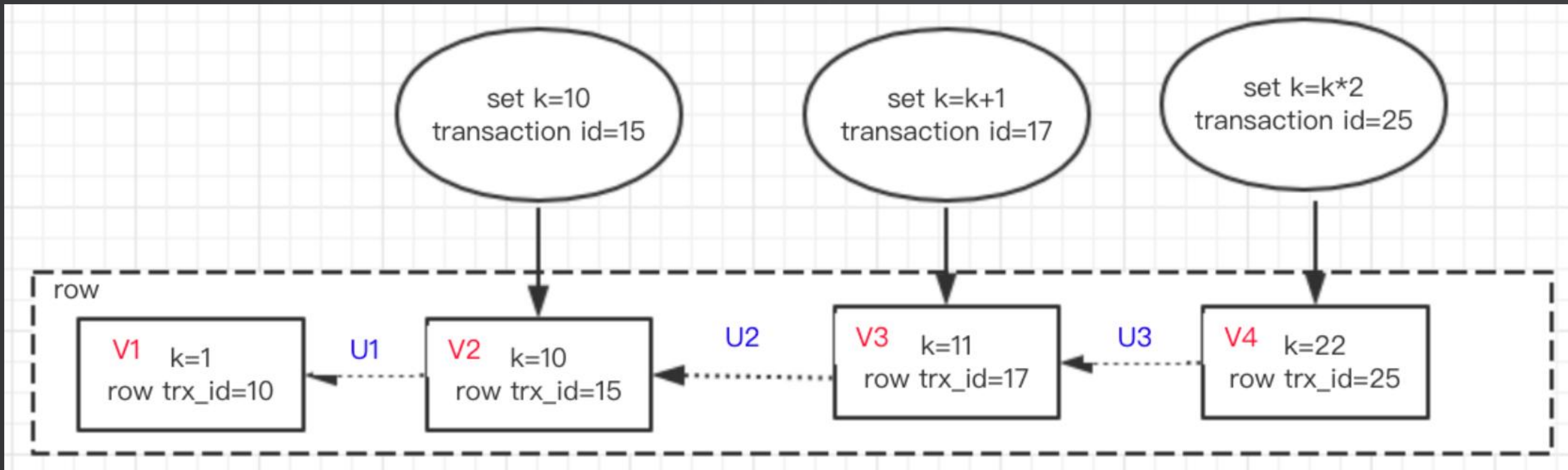
mysqldump --single-transaction

==》【官方】系统表全部改成InnoDB表



# MVCC 和 锁

select \* from t where id=N lock in share mode





# MVCC 和 锁

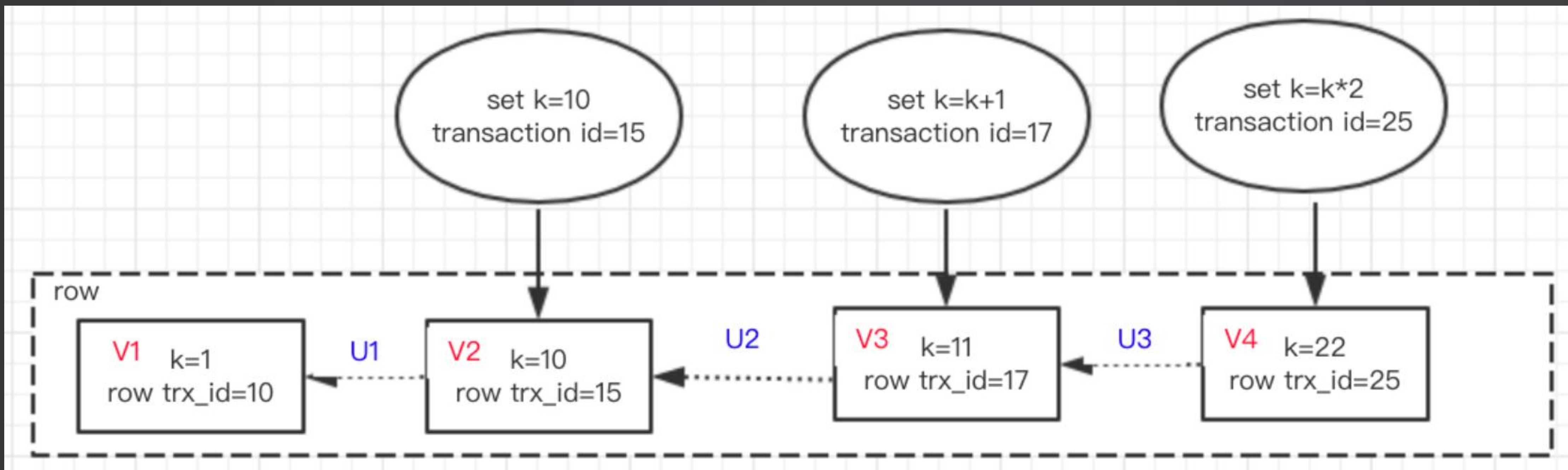
`select * from t where id=N for update;`

`select k, k*10 from t where id=N lock in share mode`

`update ... set k=k+1 where id=N`

`select benchmark(1000000,select tim from t) ;`

`time_zone = system --> +8:00`



# 课堂练习

RR隔离级别下，表t的建表结构和初始化数据如下：

```
create table t(id int primary key,c int) engine=innodb;  
insert into t values(1,1),(11,11),(21,21);
```

在会话1 执行如下语句：

```
begin;  
select * from t lock in share mode;
```

那么，会话2的以下哪些语句会被进入“等待行锁”的状态？

A: insert into t values(15,15);

B: update t set c=c+1 where id=15;

C: delete from t where id=15;

D: alter table t add d int;

# Q&A

THANKS