于延辰

InnoDb加锁分析

Backgroud

- MVCC
- Clustered and Secondary Indexes
- 2PL
- Isolation Level

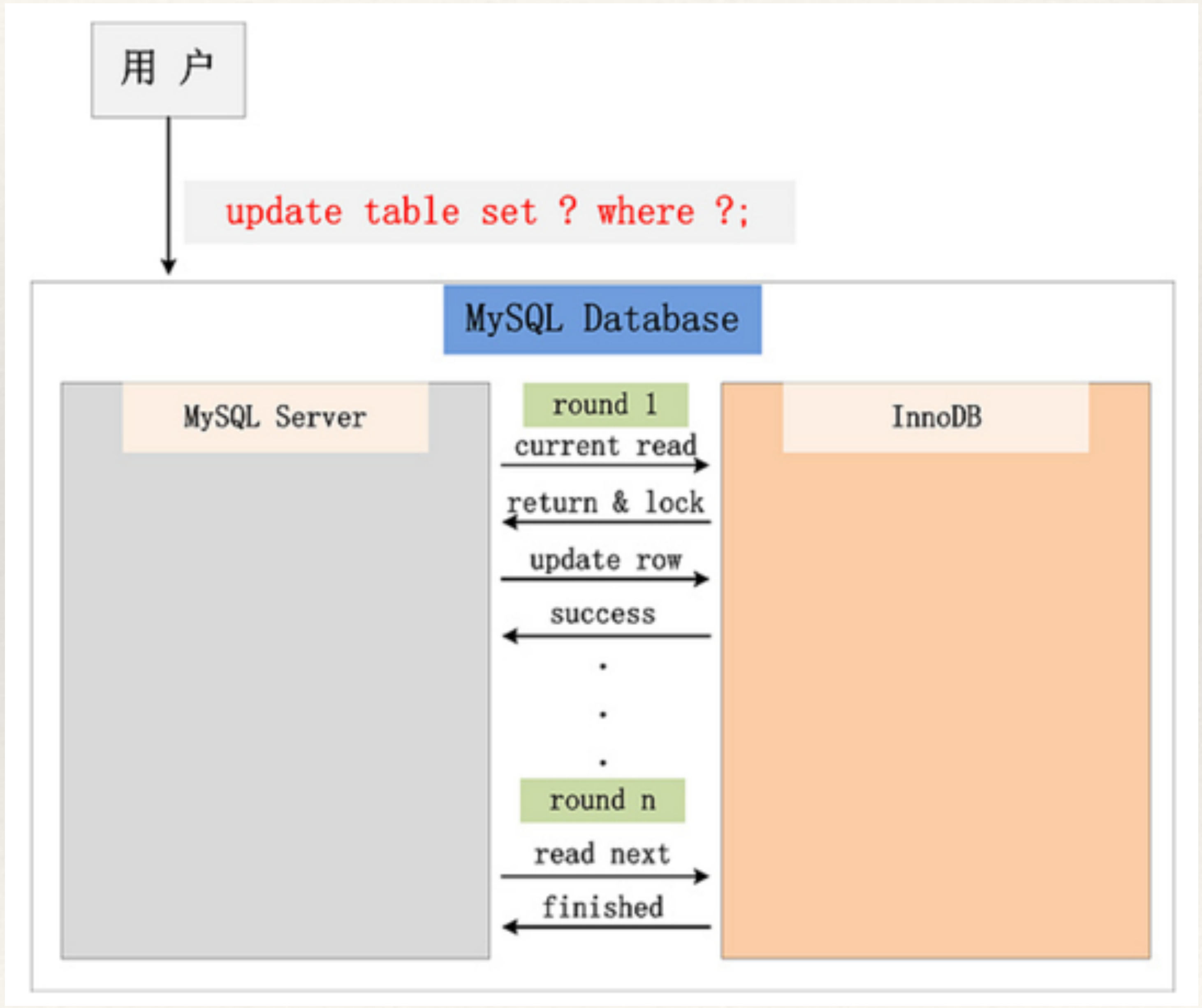
MVCC

- MVCC：Multi-Version Concurrency Control，compared Lock-Based Concurrency Control
- Advantage
  - read without lock
  - reading and writing is not conflict
- Isolation Level: RC、RR
- Read: Snapshot Read、Current Read

- Snapshot Read: read history version、unlock
    - select * from table where …
- Current Read: read latest version、lock
    - select * from table where … lock in share mode (S)
    - select * from table where … for update (X)
    - insert into table values (…) (X)
    - update table set … where … (X)
    - delete from table where … (X)

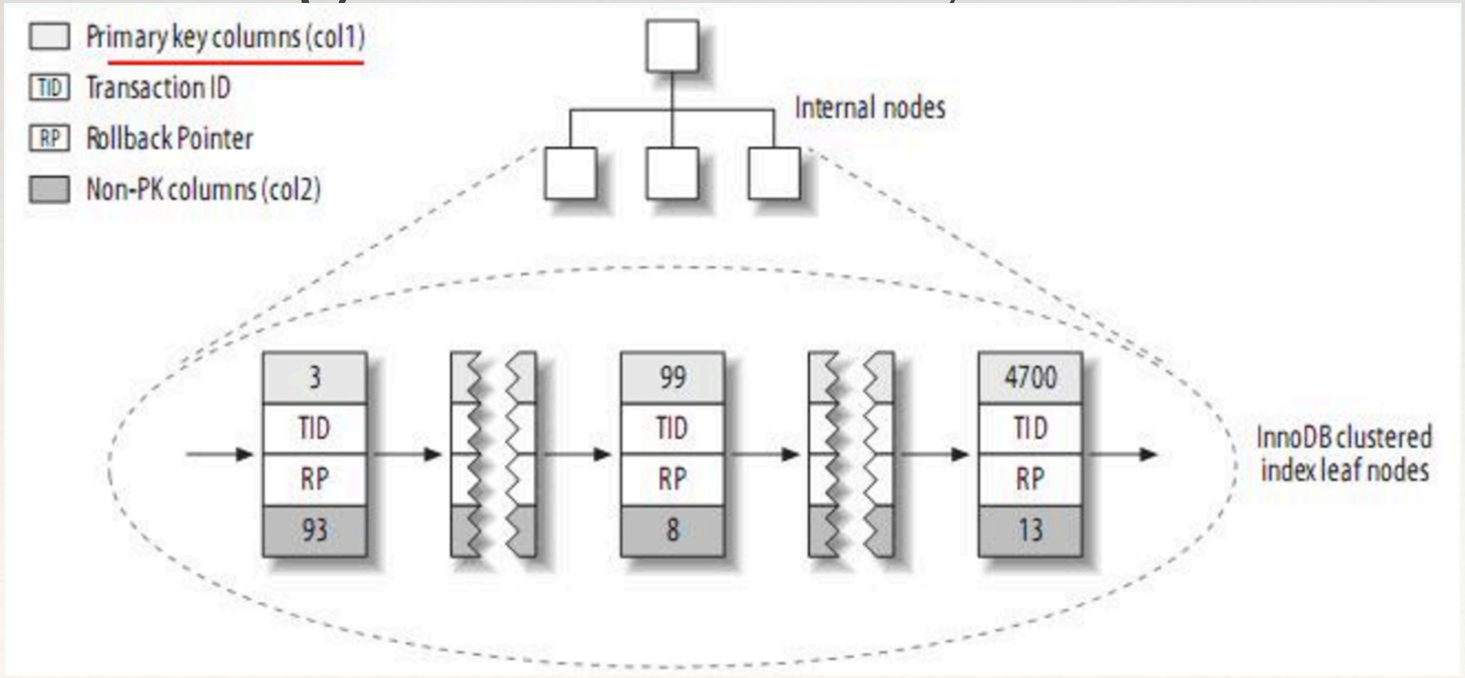
MVCC

- why update、delete、insert is current read?



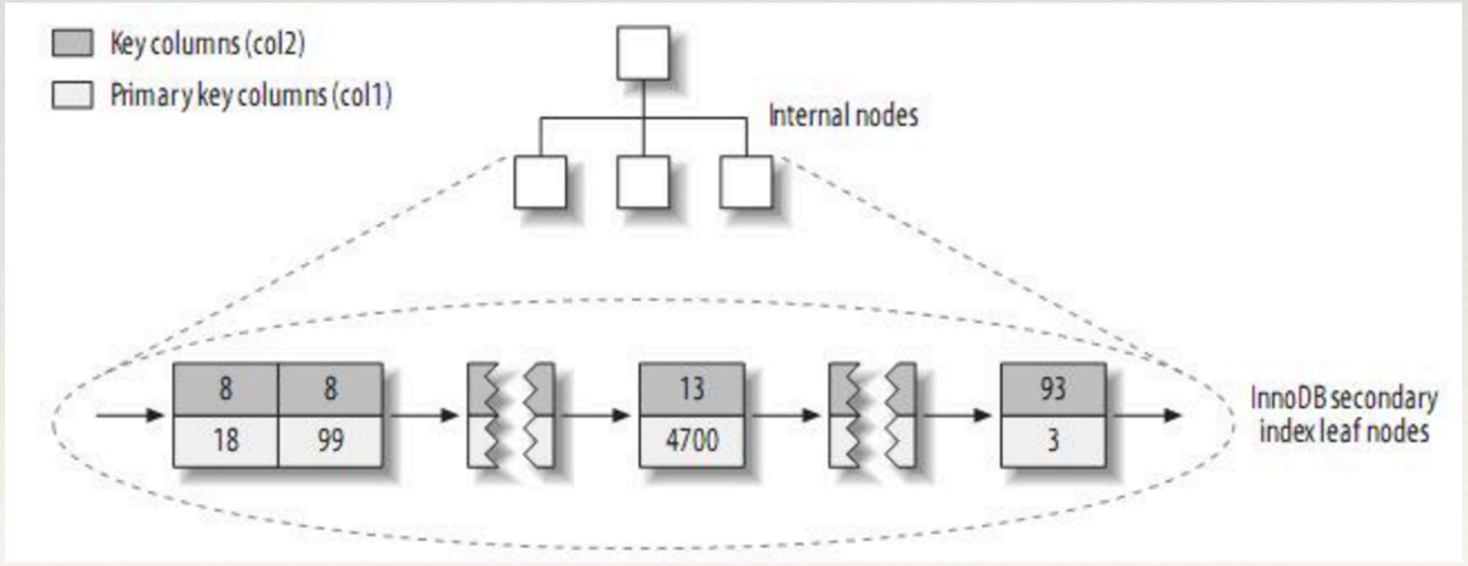
Clustered Index

- Leaf：primary key value、transaction id、roll pointer、data
- not define，locates the first UNIQUE index where all the key columns are NOT NULL
- no suitable UNIQUE index，generates a 6-byte row ID



Secondary Index

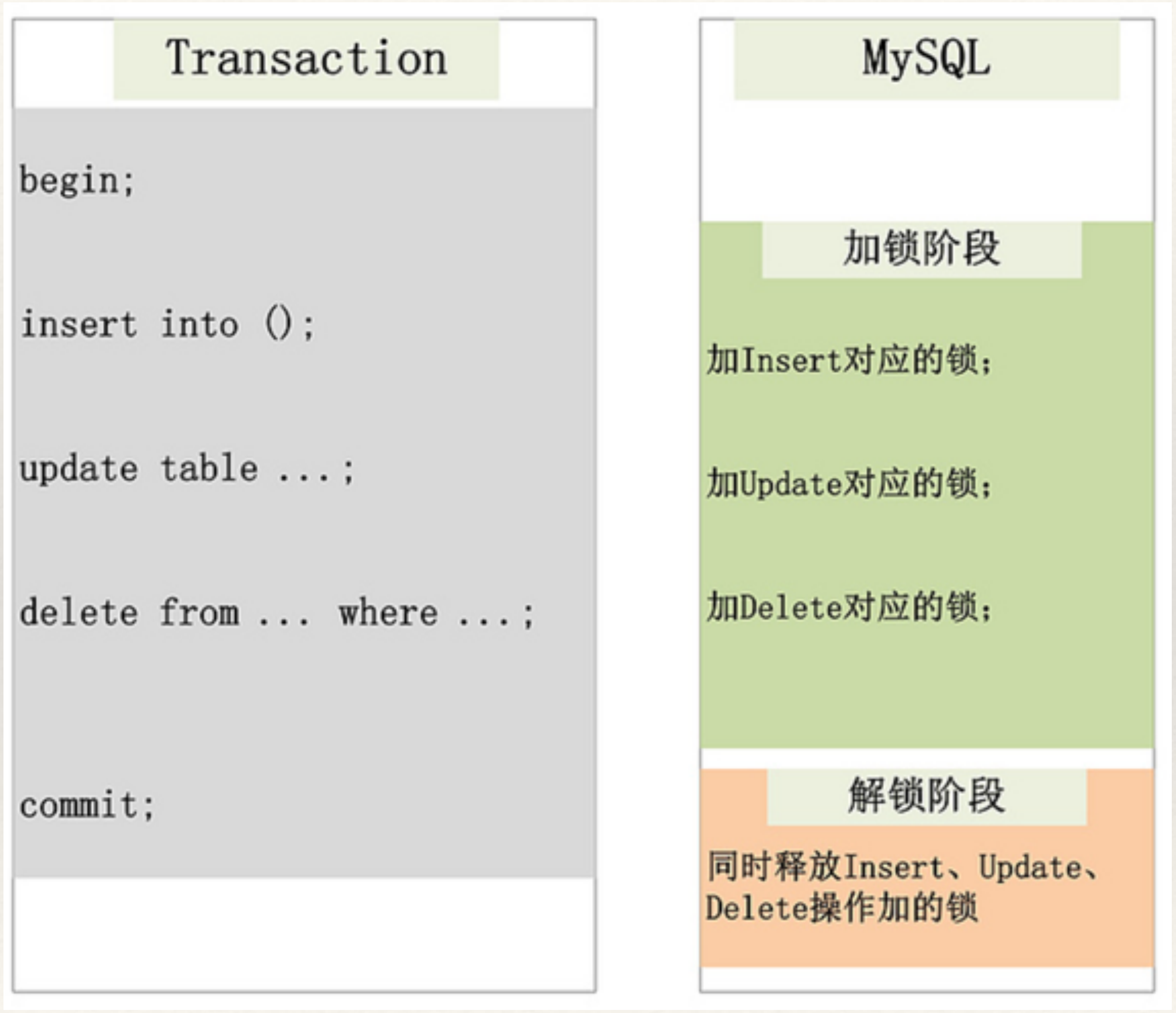
- Leaf：index Value、primary key value
- Index Scan
  - Secondary B+Tree find Primary Key
  - Clustered B+Tree find data by Primary Key



Two-phase locking

- a concurrency control method that guarantees serializability
- Expanding phase
  - locks are acquired and no locks are released
- Shrinking phase
  - locks are released and no locks are acquired.

Two-phase locking



Isolation Level

- Read Uncommited (RU)
- Read Committed (RC)
  - Snapshot Read： read latest version
  - Current Read： row lock， phantom read
- Repeatable Read (RR)
  - Snapshot Read： read transaction start version
  - Current Read： row lock + gap lock， solve phantom read
- Serializable
  - read operations are currently reading， read with read lock (S lock)
  - write with write lock (X lock)

Simple SQL Lock Realize Analyse

- SQL1： select * from t1 where id = 10;
- SQL2： delete from t1 where id = 10;
- Precondition
  - id is primary key、secondary index、unique index？
  - isolation level?
  - index scan？ table scan?

Simple SQL Lock Realize Analyse

- id PK + RC
- id Unique Index + RC
- id NonUnique Index + RC
- id Indexless + RC
- id PK + RR
- id Unique Index + RR
- id NonUnique Index + RR
- id Indexless + RR

PK + RC

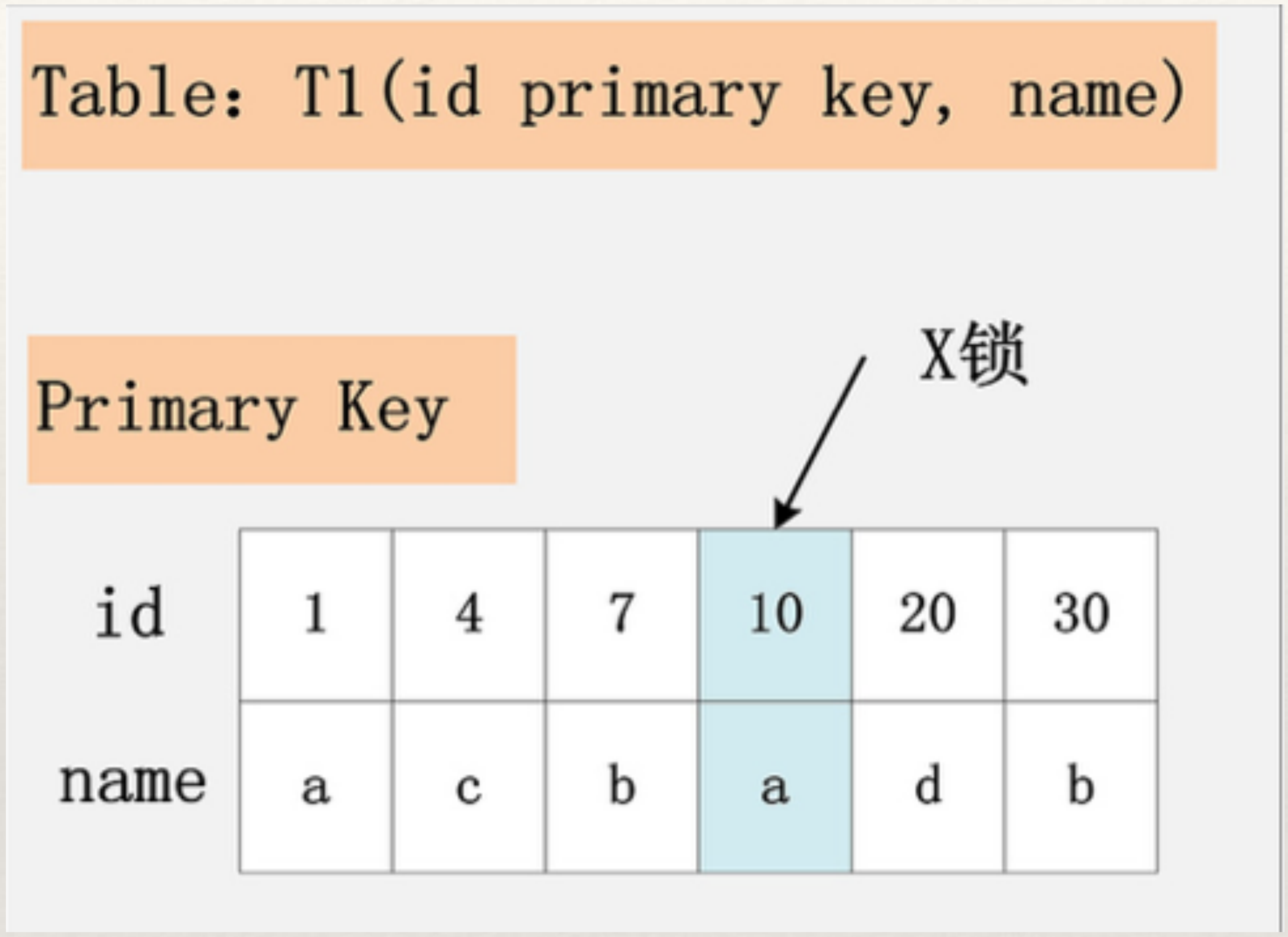


Table: T1(id primary key, name)

Primary Key

id	1	4	7	10	20	30
name	a	c	b	a	d	b

- PK id=10 record adds X lock

Unique Index + RC

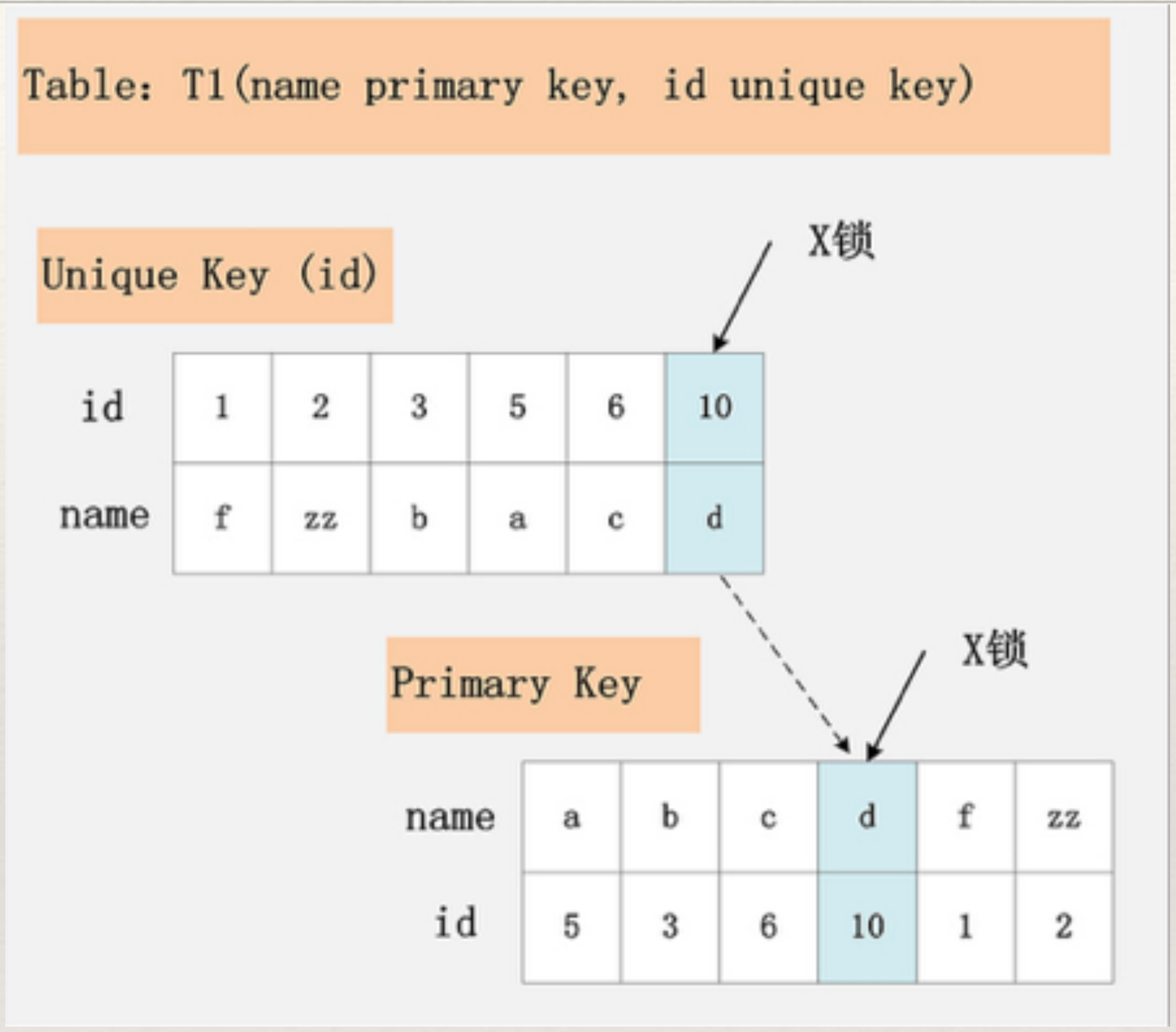


Table: T1(name primary key, id unique key)

Unique Key (id)

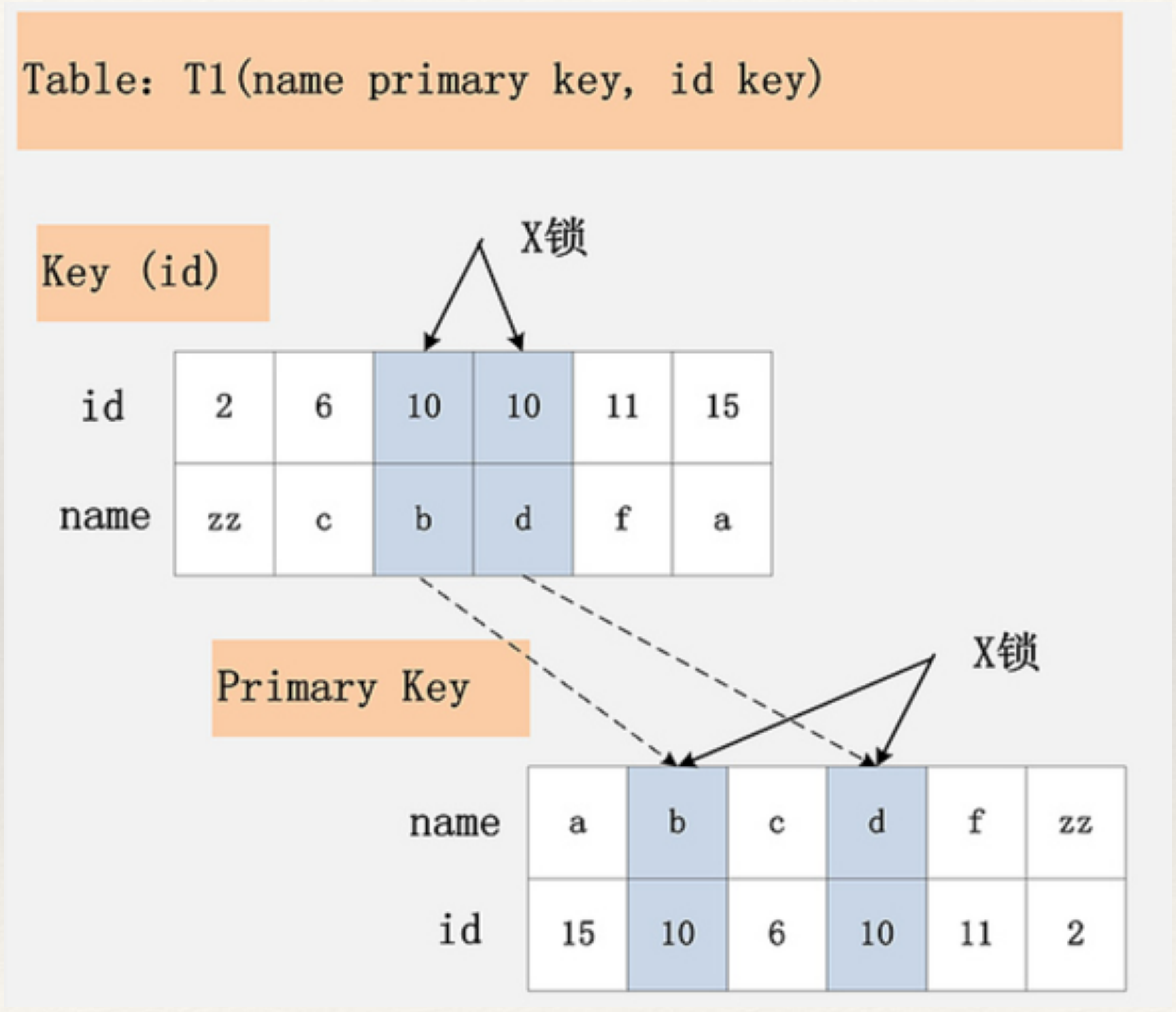
id	1	2	3	5	6	10
name	f	zz	b	a	c	d

Primary Key

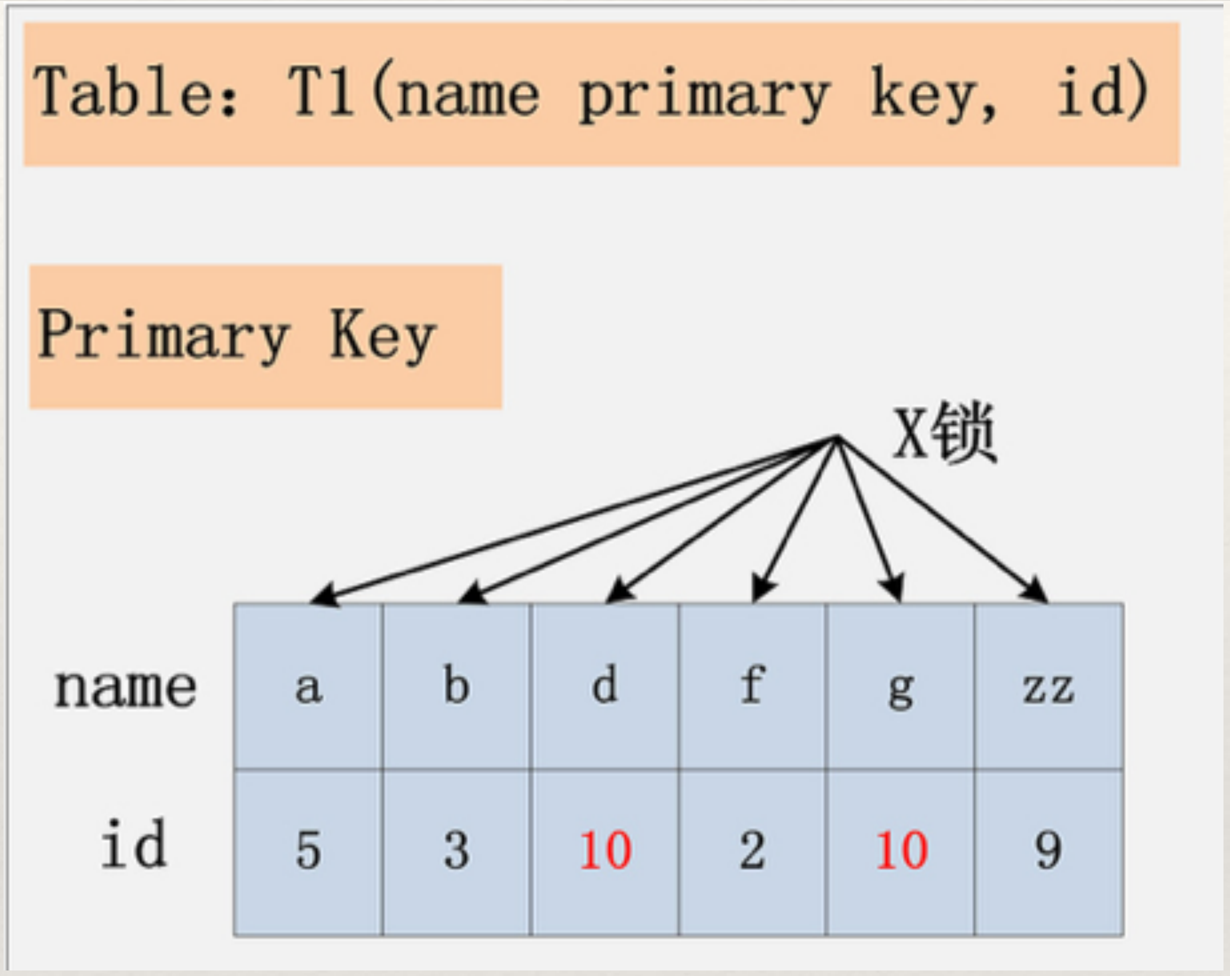
name	a	b	c	d	f	zz
id	5	3	6	10	1	2

- why PK is locked?
- update t1 set id = 100 where name = ‘d’
- same record, delete and update serial execution

NonUnique Index + RC

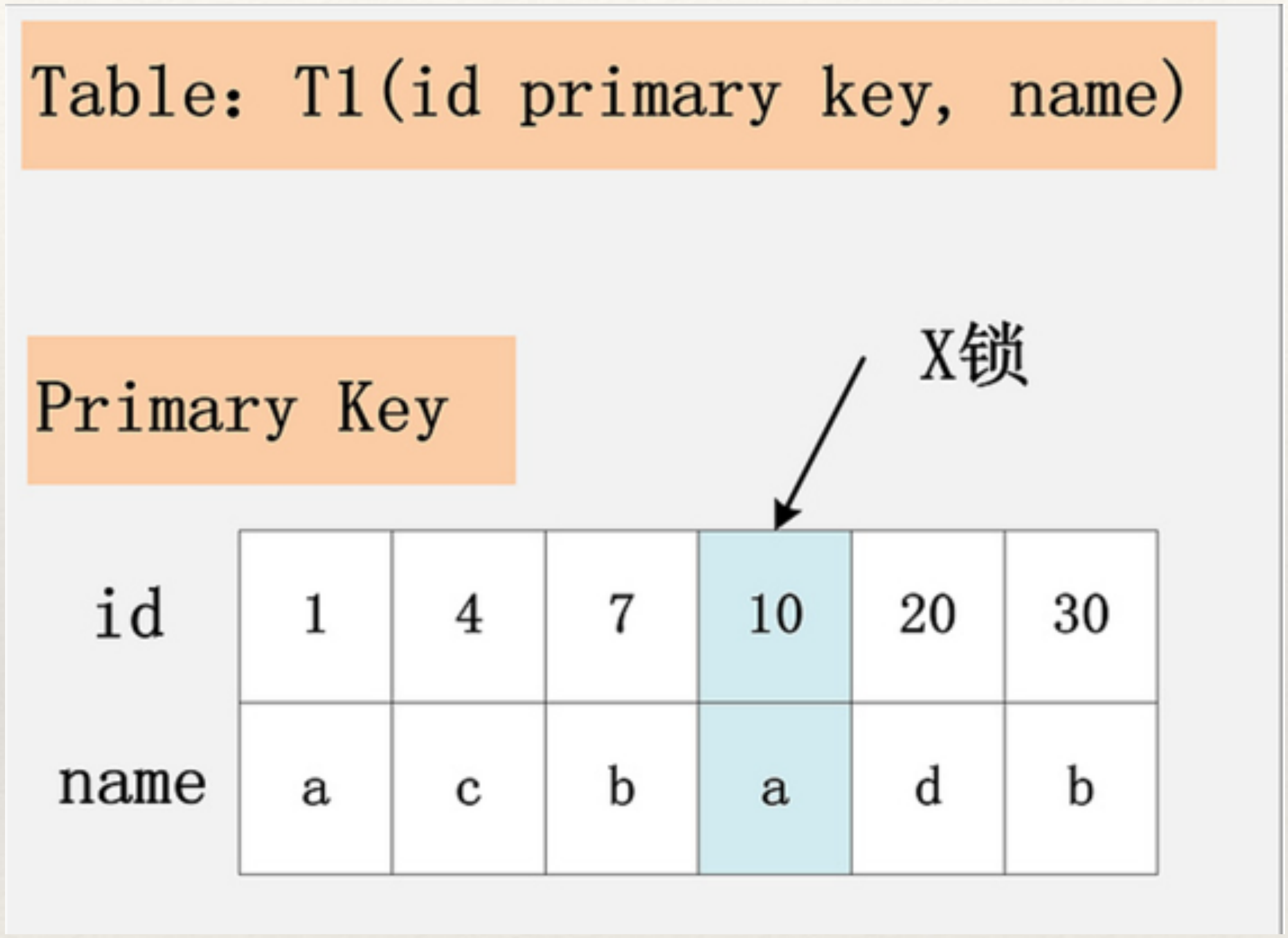


No Index + RC



- table scan by clustered index
- MySQL server will filter data， mysql optimize， semi-consistent read， go against the 2PL
- each record is locked and unlocked

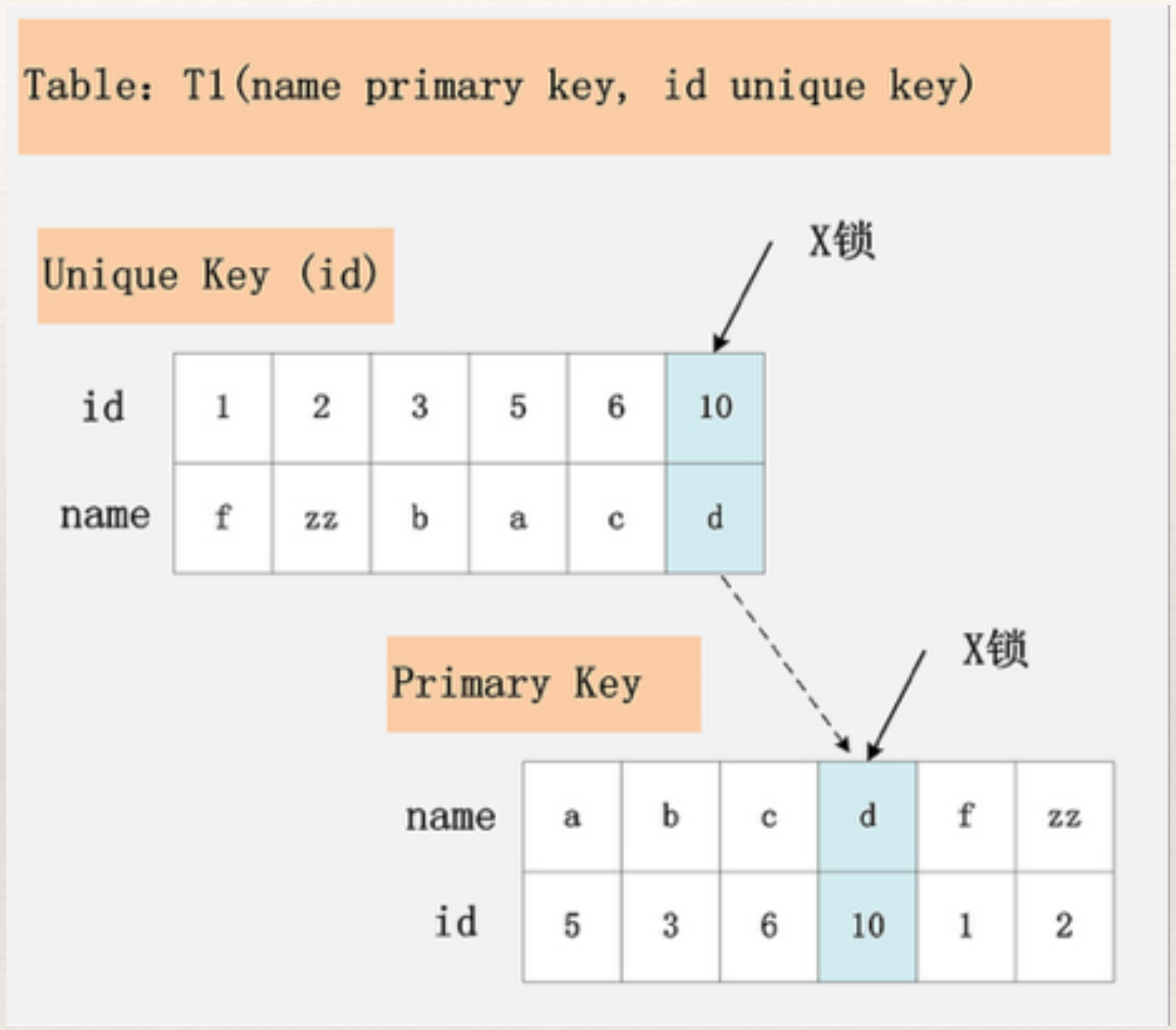
PK + RR



id	1	4	7	10	20	30
name	a	c	b	a	d	b

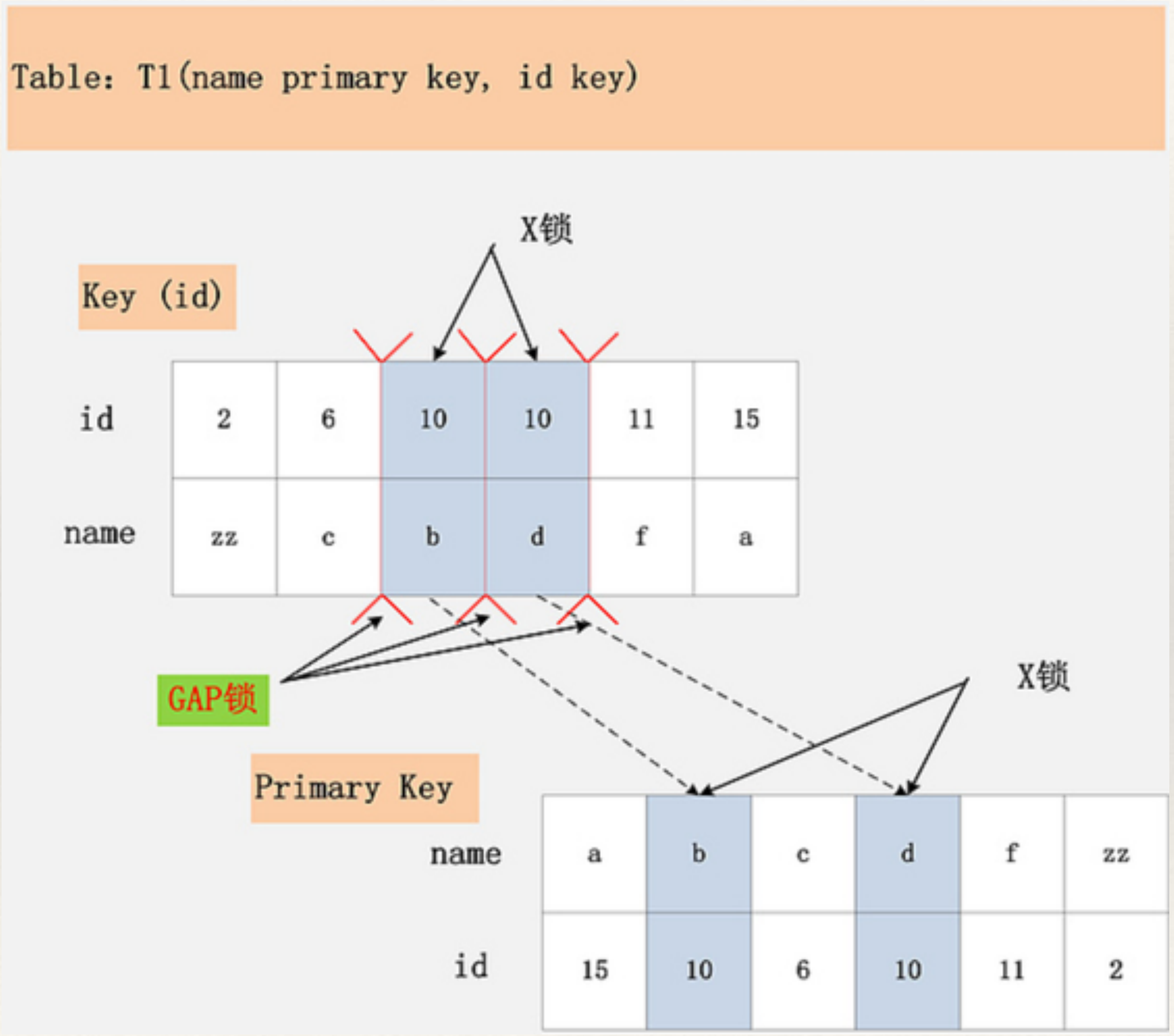
- the same with PK + RC

Unique Index + RR



- the same with Unique Index + RC

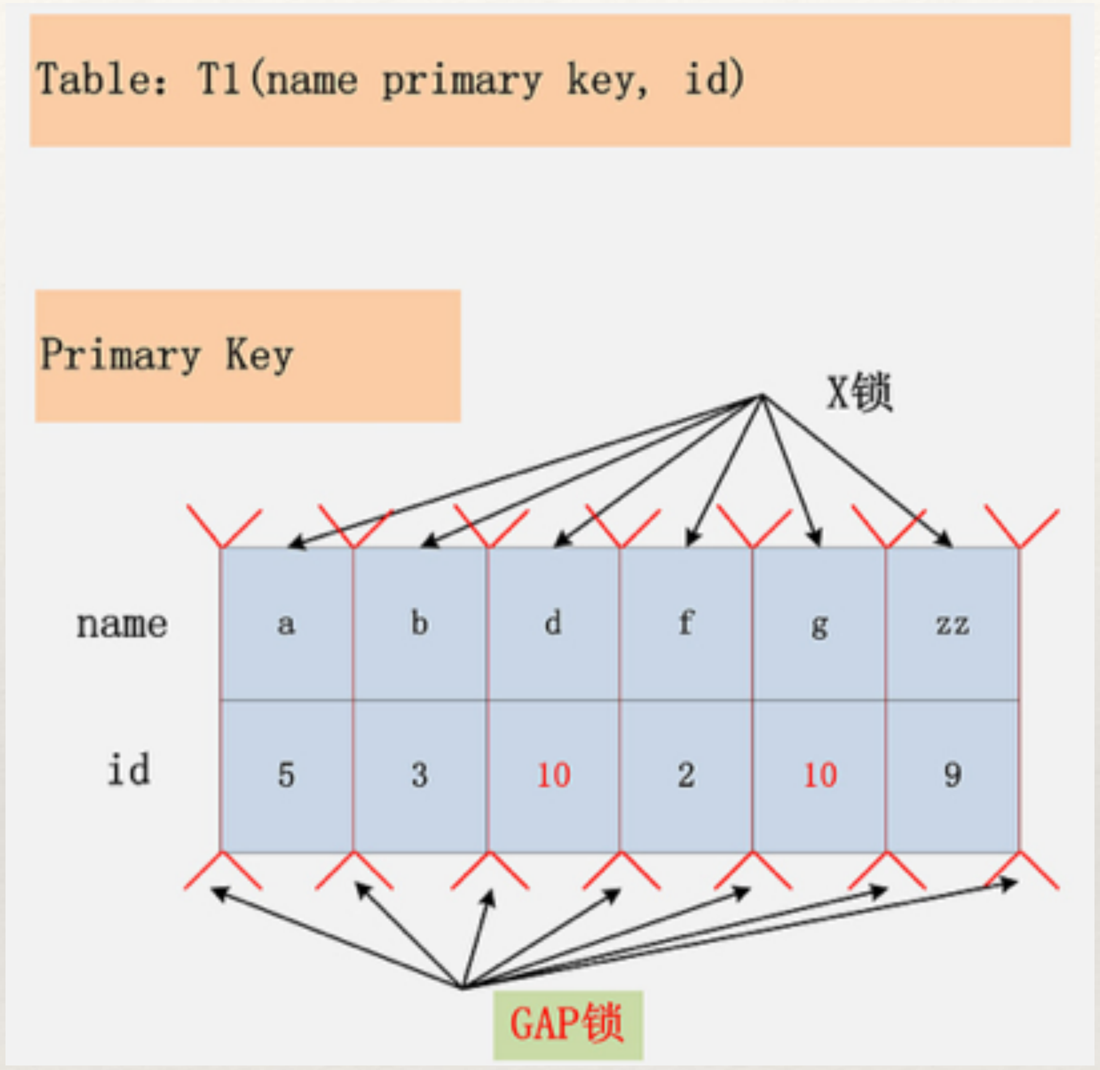
NonUnique Index + RR



NonUnique Index + RR

- what is phantom read?
- when needs the gap lock?
  - equality query
- considering :
  - select * from t1 where id = 10 for update
  - If not have records meeting the condition
    - id PK + RR ?
    - id Unique Key + RR ?

No Index + RR



- table scan
- semi-consistent read

Complicated SQL Lock Realize Analyse

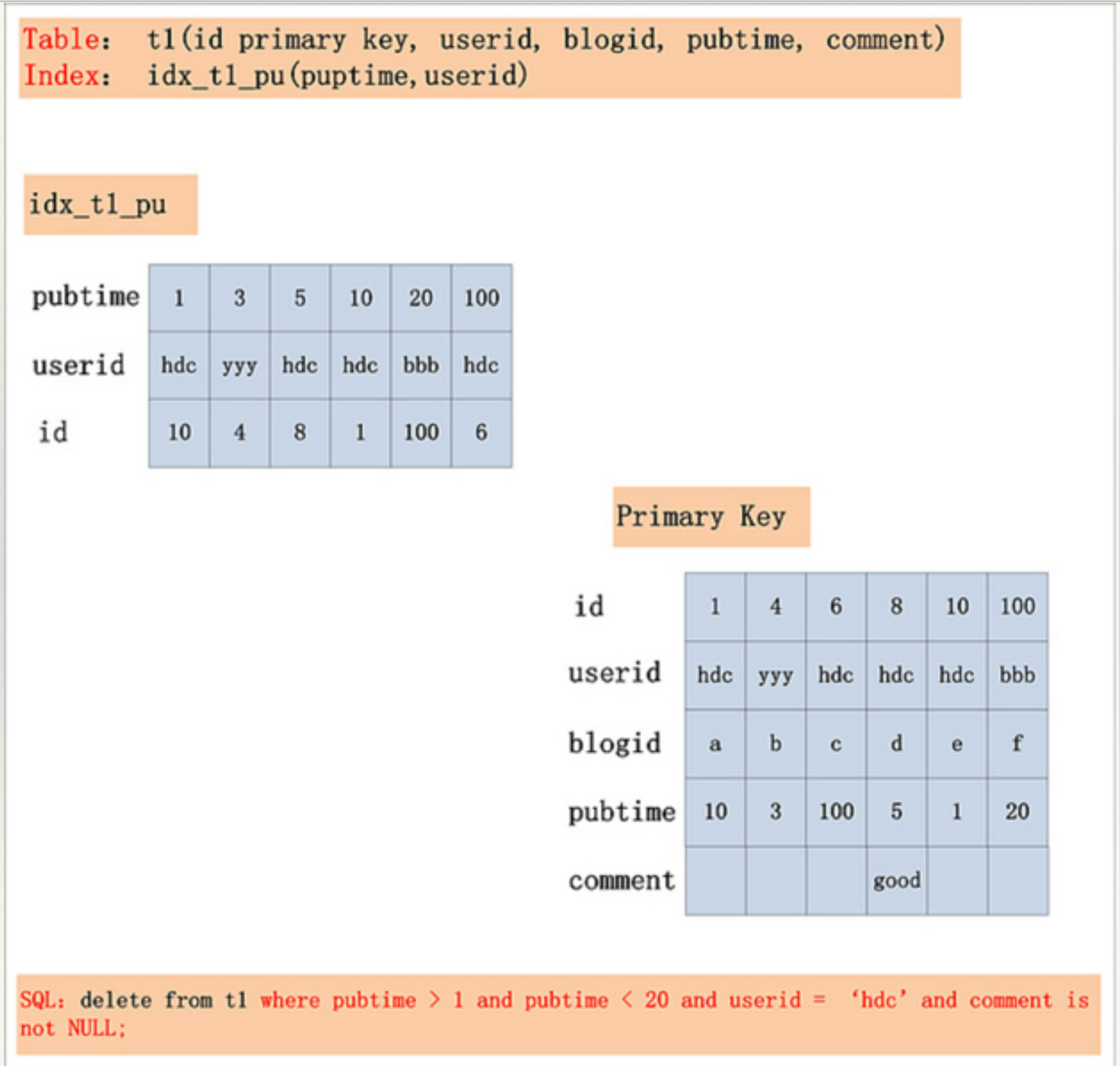


Table: t1(id primary key, userid, blogid, pubtime, comment)
Index: idx_t1_pu(puptime,userid)

idx_t1_pu

pubtime	1	3	5	10	20	100
userid	hdc	yyy	hdc	hdc	bbb	hdc
id	10	4	8	1	100	6

Primary Key

id	1	4	6	8	10	100
userid	hdc	yyy	hdc	hdc	hdc	bbb
blogid	a	b	c	d	e	f
pubtime	10	3	100	5	1	20
comment				good		

```
SQL: delete from t1 where pubtime > 1 and pubtime < 20 and userid = 'hdc' and comment is
not NULL;
```

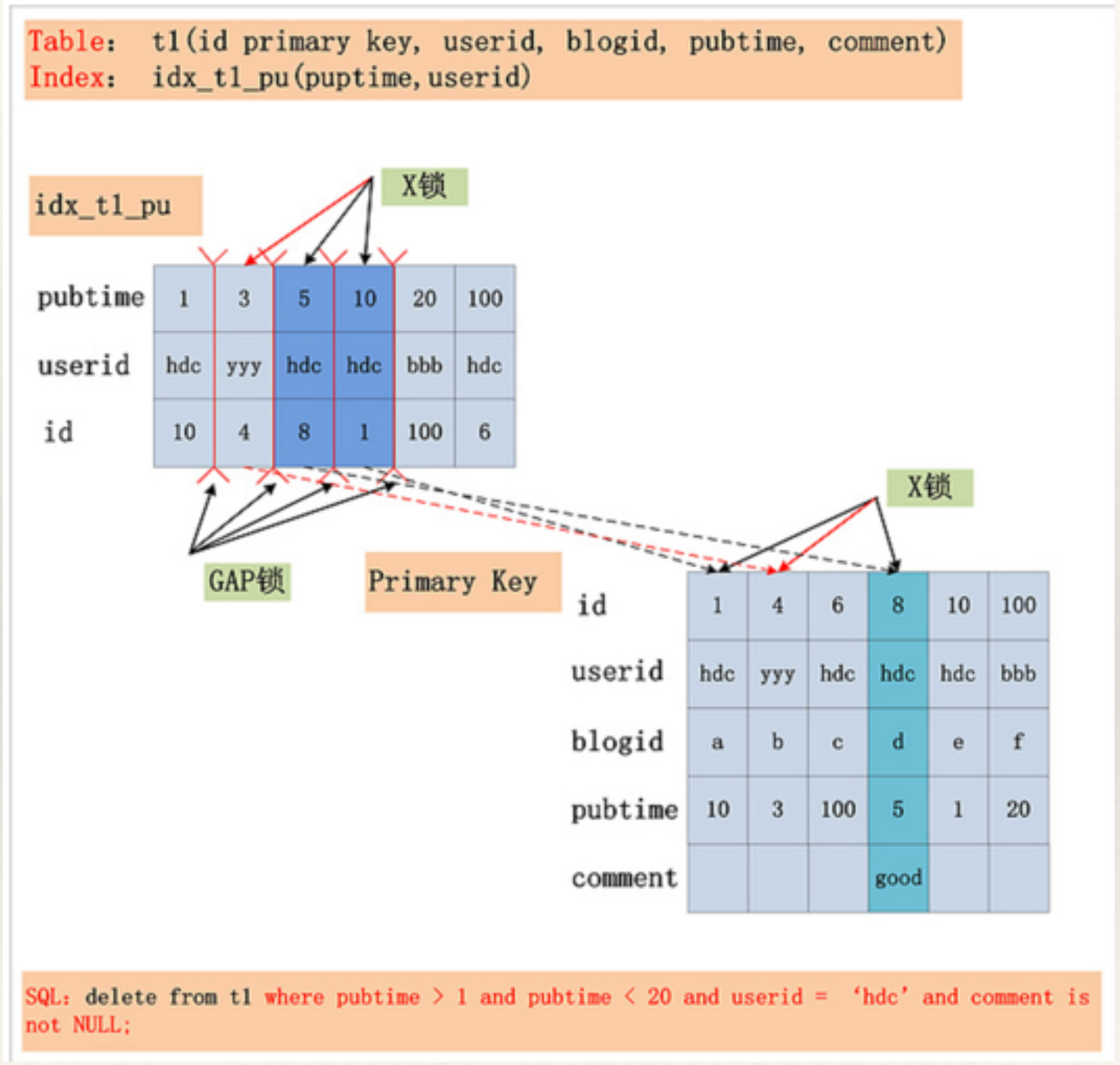
where conditions extraction

- index key
  - index first key, index last key
  - locates the index searching range
- index filter
  - each record filters
  - MySQL 5.6 brings in Index Condition Pushdown， before not distinguishes index filter and table filter，
- table filter
  - back to the clustered index to read the data

Complicated SQL Lock Realize Analyse

- Index key
    - pubtime > 1 and puptime < 20
    - idx_t1_pu
- Index Filter
    - userid = ‘hdc’
    - idx_t1_pu filter
- Table Filter
    - comment is not NULL
    - clustered index filter

Complicated SQL Lock Realize Analyse



```
Table:  t1(id primary key, userid, blogid, pubtime, comment)
Index:  idx_t1_pu(puptime,userid)
```

idx_t1_pu

pubtime	1	3	5	10	20	100
userid	hdc	yyy	hdc	hdc	bbb	hdc
id	10	4	8	1	100	6

Primary Key

id	1	4	6	8	10	100
userid	hdc	yyy	hdc	hdc	hdc	bbb
blogid	a	b	c	d	e	f
pubtime	10	3	100	5	1	20
comment				good		

```
SQL: delete from t1 where pubtime > 1 and pubtime < 20 and userid = 'hdc' and comment is not NULL;
```

References

- https://dev.mysql.com/doc
- http://hedengcheng.com/?p=771#_Toc374698307
- http://hedengcheng.com/?p=220
- http://hedengcheng.com/?p=577

- Thanks !