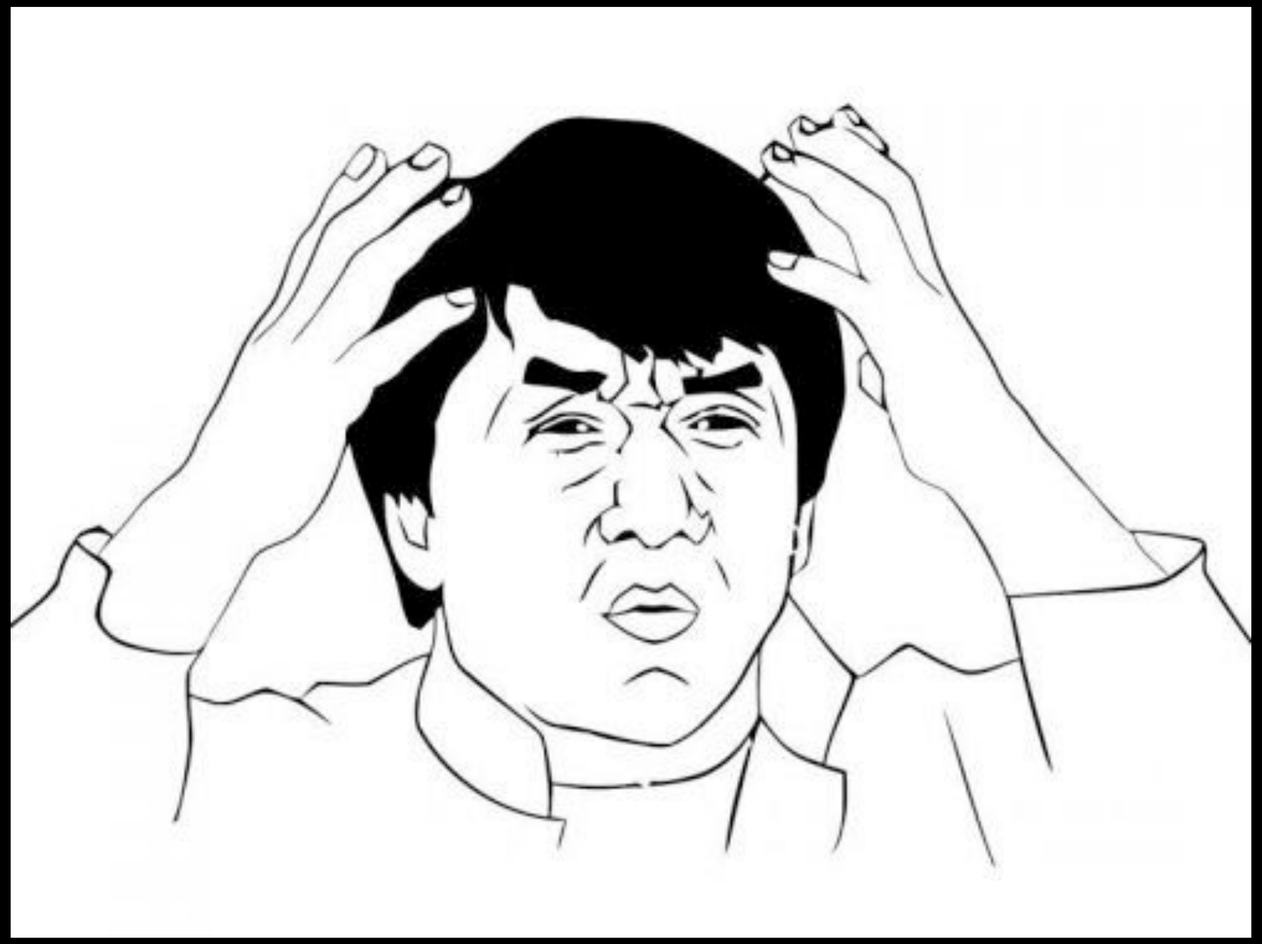
MySQL Lock



Deadlock & Slow query

MySQL Lock

- Isolation Level
  - Lock
  - MVCC

“isolation level is the setting that fine-tunes the balance between performance and reliability, consistency, and reproducibility of results when multiple transactions are making changes and performing queries at the same time.”

- https://dev.mysql.com/doc/refman/8.0/en/innodb-transaction-isolation-levels.html

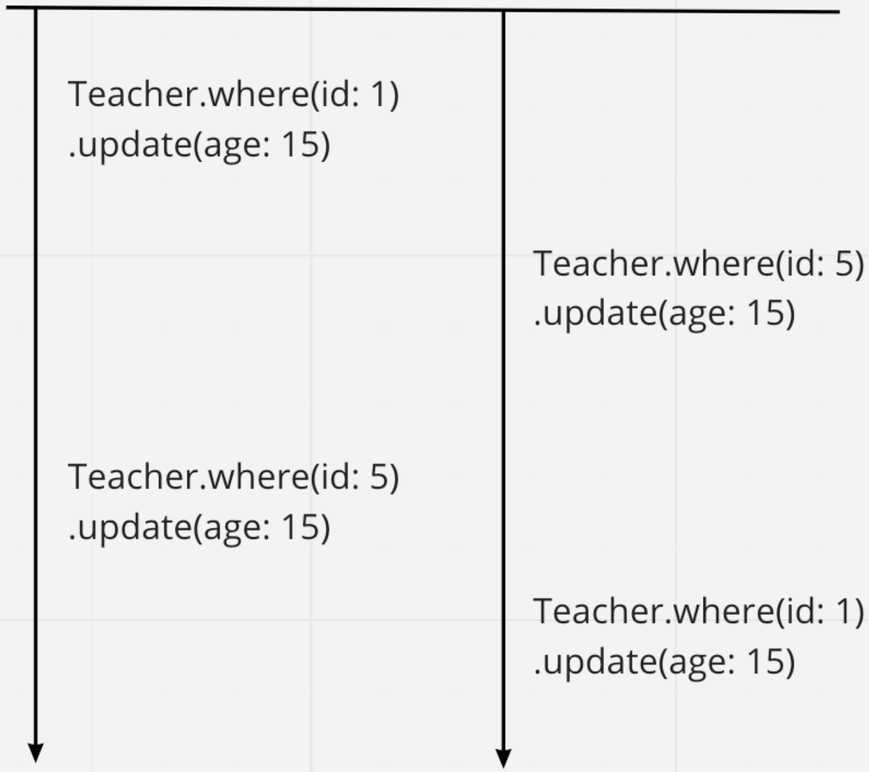
Agenda

- MySQL Lock
  - Lock and Index
  - Deadlock
  - Deadlock Error Message
  - Slow Query
- Real Case
- Conclusion

Precondition

- MySQL 5.7
- Repeatable Read
- Table Teachers
  - Indice:
    - Primary key: id
    - Unique Index: name
    - Secondary Index: age

X Warm Up 1



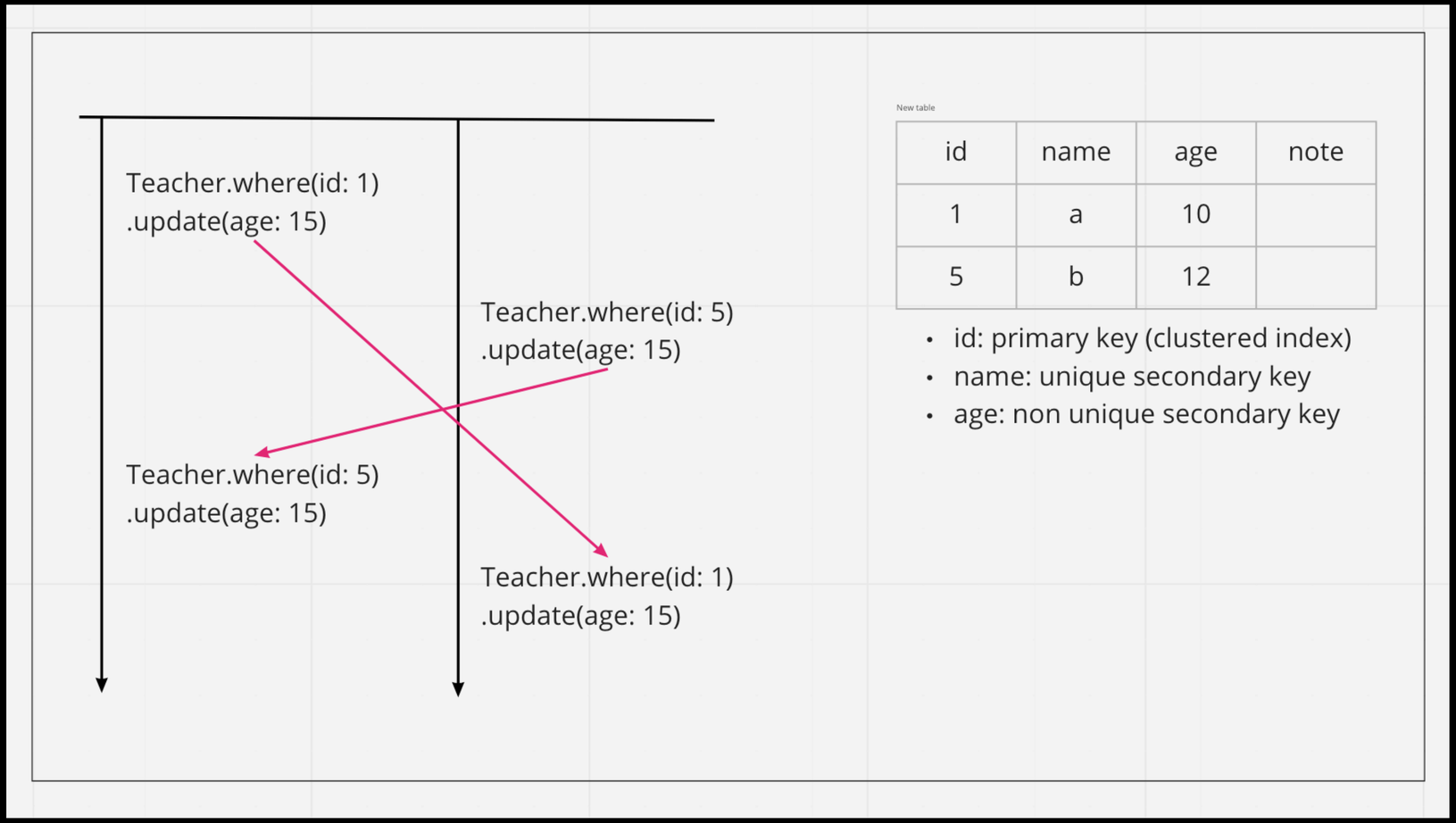
New table

id	name	age	note
1	a	10	
5	b	12	

- id: primary key (clustered index)
- name: unique secondary key
- age: non unique secondary key

Deadlock

- Mutual Exclusive
- Hold and Wait
- No Preemption
- Circular Wait



How to Debug DeadLock

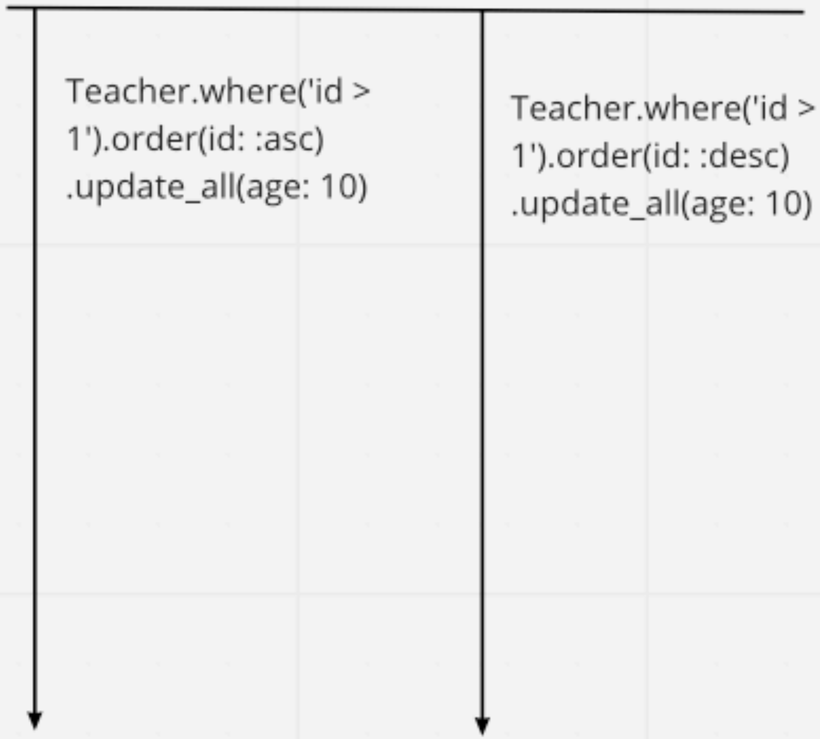
- Deadlock Log
  - innodb_deadlock_detect: 是否偵測 deadlock
  - innodb_print_all_deadlocks: 如果要全部 log 到 error.log 需要打開 flag
  - 否則只能看到最後一筆 deadlock
    > SHOW ENGINE INNODB STATUS

How to Debug DeadLock

```
------------------------
2022-04-21 10:36:45 0x40de538700
*** (1) TRANSACTION:
TRANSACTION 8115, ACTIVE 3 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 3 lock struct(s), heap size 1136, 2 row lock(s), undo log entries 1
MySQL thread id 827, OS thread handle 278607652608, query id 11483 172.17.0.1 root
updating
UPDATE `teachers` SET `teachers`.`age` = 6 WHERE `teachers`.`id` = 1
*** (1) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 227 page no 3 n bits 72 index PRIMARY of table
`test`.`teachers` trx id 8115 lock_mode X locks rec but not gap waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 6; compact format; info bits 0
 0: len 8; hex 8000000000000001; asc        ;;
 1: len 6; hex 000000001fb2; asc       ;;
 2: len 7; hex 53000001cc295d; asc S    )];;
 3: len 3; hex 616161; asc aaa;;
 4: len 4; hex 8000000f; asc     ;;
 5: SQL NULL;


------------
```

```
*** (2) TRANSACTION:
TRANSACTION 8114, ACTIVE 3 sec starting index read
mysql tables in use 1, locked 1
3 lock struct(s), heap size 1136, 2 row lock(s), undo log entries 1
MySQL thread id 828, OS thread handle 278607922944, query id 11482 172.17.0.1 root
updating
UPDATE `teachers` SET `teachers`.`age` = 10 WHERE `teachers`.`id` = 5
*** (2) HOLDS THE LOCK(S):
RECORD LOCKS space id 227 page no 3 n bits 72 index PRIMARY of table `test`.`teachers`
trx id 8114 lock_mode X locks rec but not gap
Record lock, heap no 2 PHYSICAL RECORD: n_fields 6; compact format; info bits 0
 0: len 8; hex 8000000000000001; asc        ;;
 1: len 6; hex 000000001fb2; asc       ;;
 2: len 7; hex 53000001cc295d; asc S    )];;
 3: len 3; hex 616161; asc aaa;;
 4: len 4; hex 8000000f; asc     ;;
 5: SQL NULL;

*** (2) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 227 page no 3 n bits 72 index PRIMARY of table `test`.`teachers`
trx id 8114 lock_mode X locks rec but not gap waiting
Record lock, heap no 3 PHYSICAL RECORD: n_fields 6; compact format; info bits 0
 0: len 8; hex 8000000000000005; asc        ;;
 1: len 6; hex 000000001fb3; asc       ;;
 2: len 7; hex 54000001ca2ac9; asc T    * ;;
 3: len 3; hex 626262; asc bbb;;
 4: len 4; hex 80000010; asc     ;;
 5: SQL NULL;

*** WE ROLL BACK TRANSACTION (2)
```

X Warm Up 2



New table

id	name	age	note
1	a	10	
5	b	12	

- id: primary key (clustered index)
- name: unique secondary key
- age: non unique secondary key

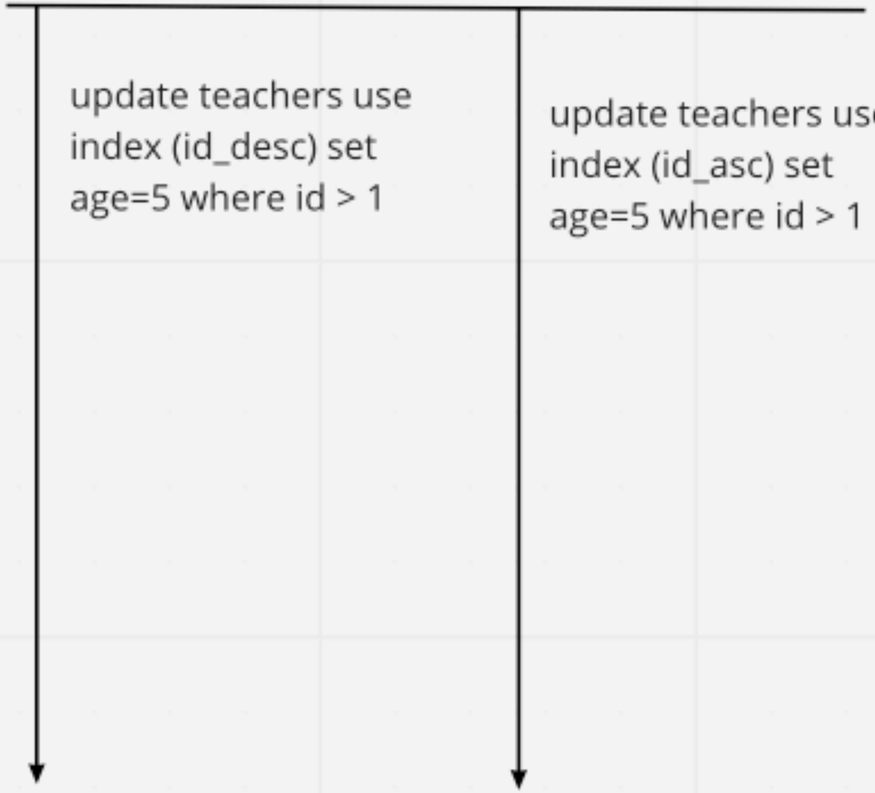
Row Lock - One By One

- Row locks are acquired one by one
  - ORDER BY
  - Index ordering (MySQL 8.0+)
- Follow index orders

> “Transaction an UPDATE statement includes an ORDER BY clause,
> the rows are updated in the order specified by the clause.”
>
> - https://dev.mysql.com/doc/refman/5.7/en/update.html

> “A *key_part* specification can end with ASC or DESC. These keywords are permitted
> for future extensions for specifying ascending or descending index value storage.
> Currently, they are parsed but ignored”
>
> - https://dev.mysql.com/doc/refman/5.7/en/create-index.html

* Warm Up 2 - 1

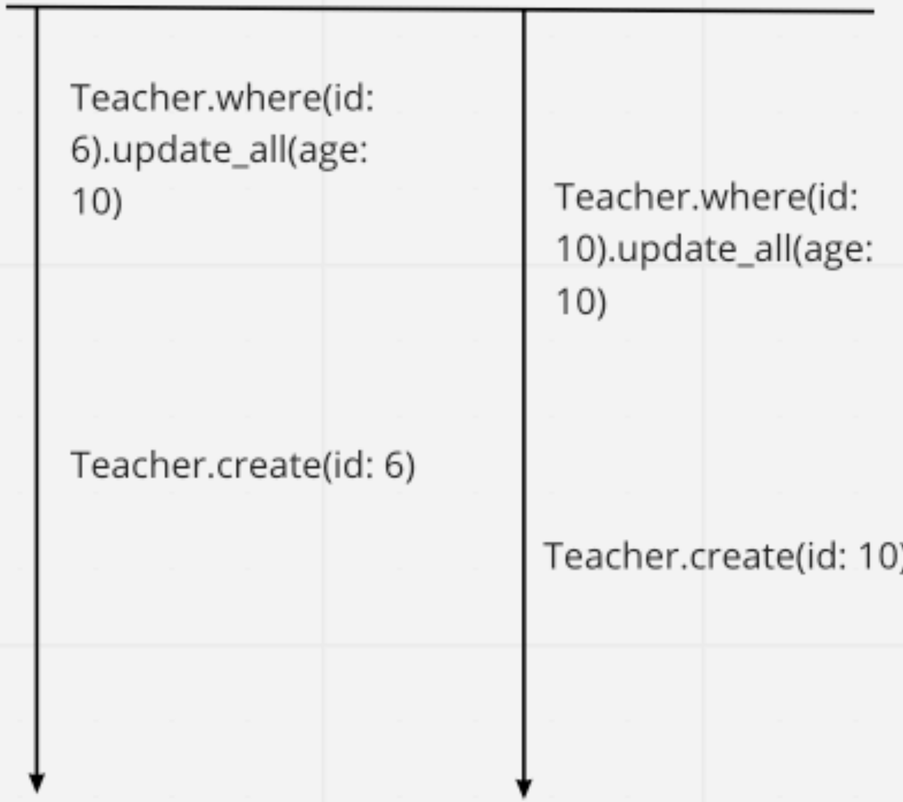


New table

id	name	age	note
1	a	10	
5	b	12	

- id: primary key (clustered index)
- name: unique secondary key
- age: non unique secondary key
- id_asc: 'create unique index id_asc on teachers (id ASC)'
- id_desc: 'create unique index id_desc on teachers (id DESC)

X Warm Up 3

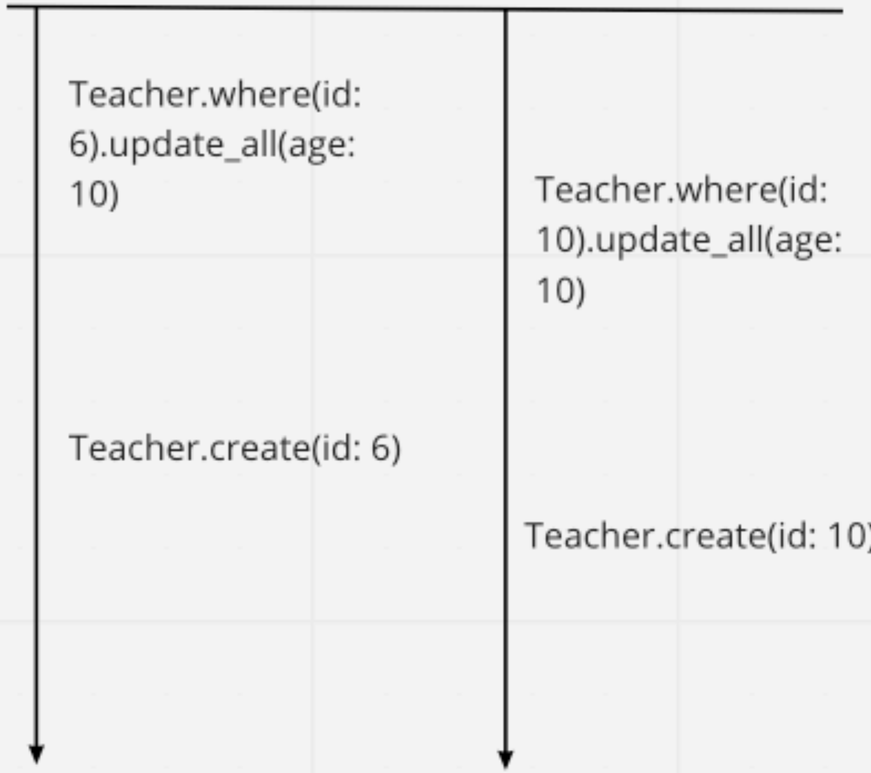


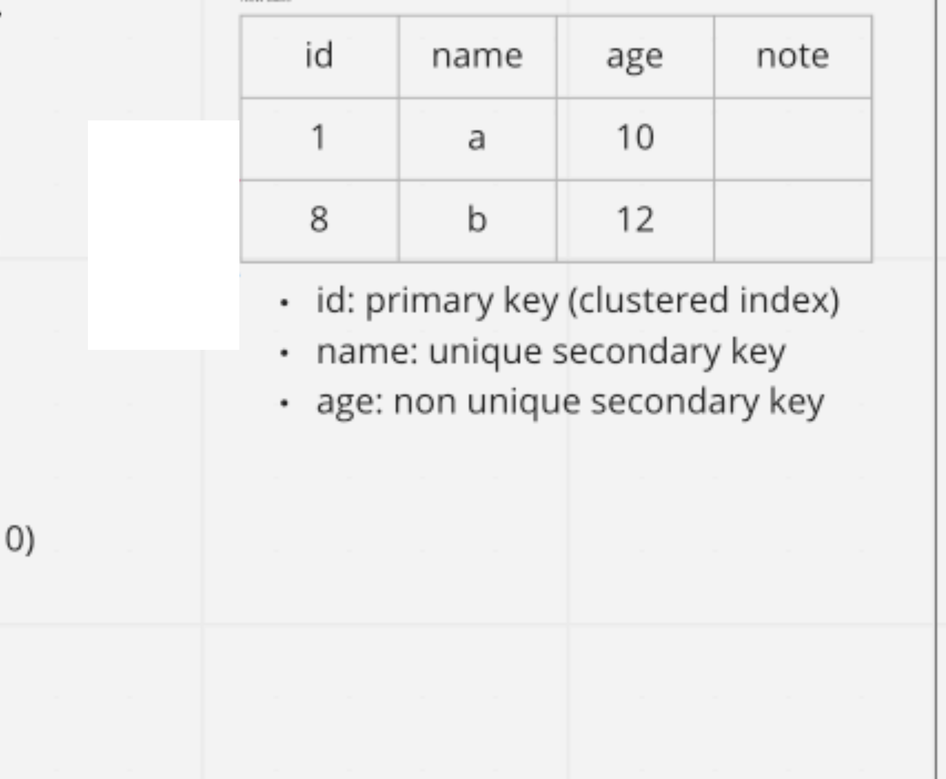
New table

id	name	age	note
1	a	10	
5	b	12	

- id: primary key (clustered index)
- name: unique secondary key
- age: non unique secondary key

O Warm Up 3 - 1





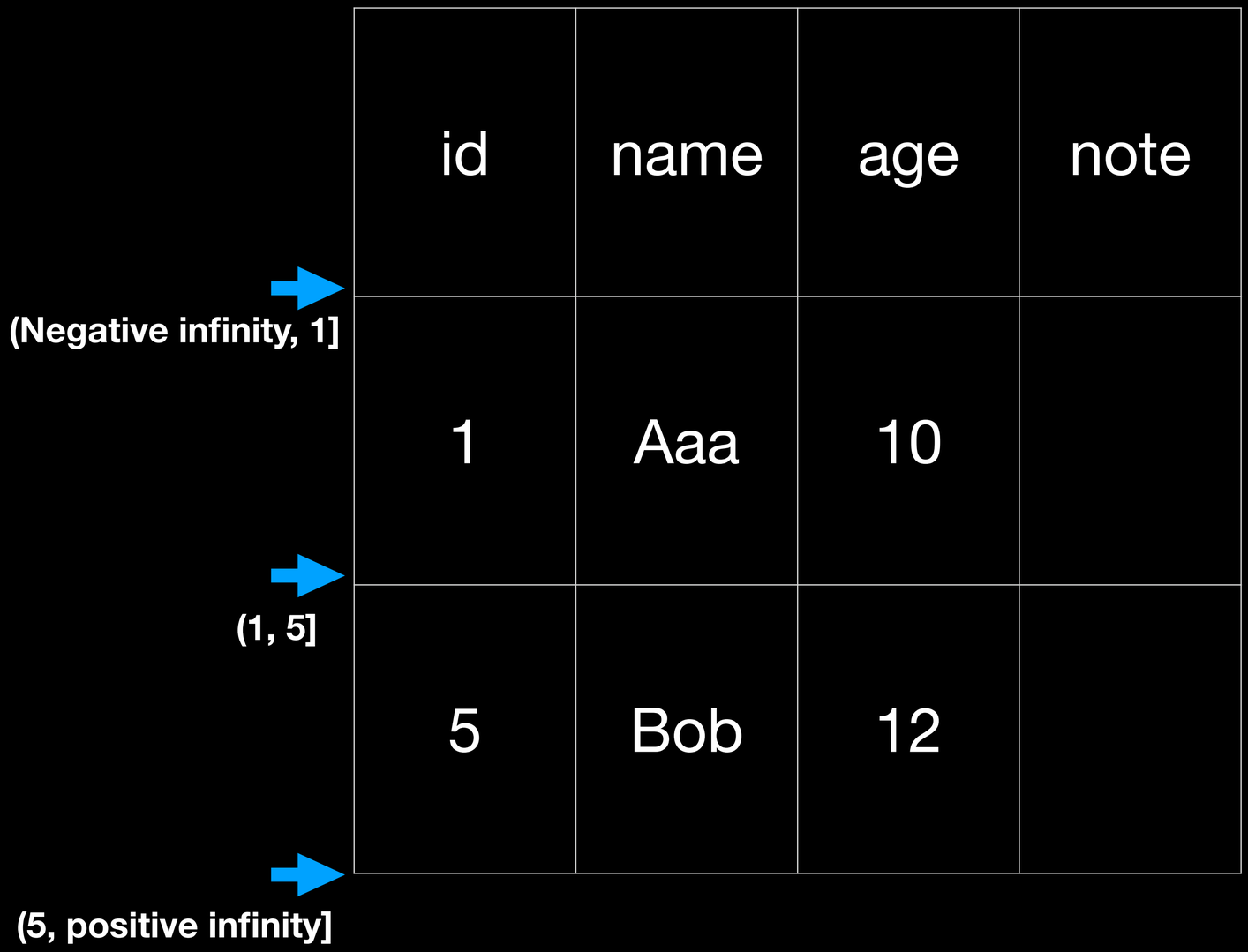
New table

id	name	age	note
1	a	10	
8	b	12	

- id: primary key (clustered index)
- name: unique secondary key
- age: non unique secondary key

MySQL Lock

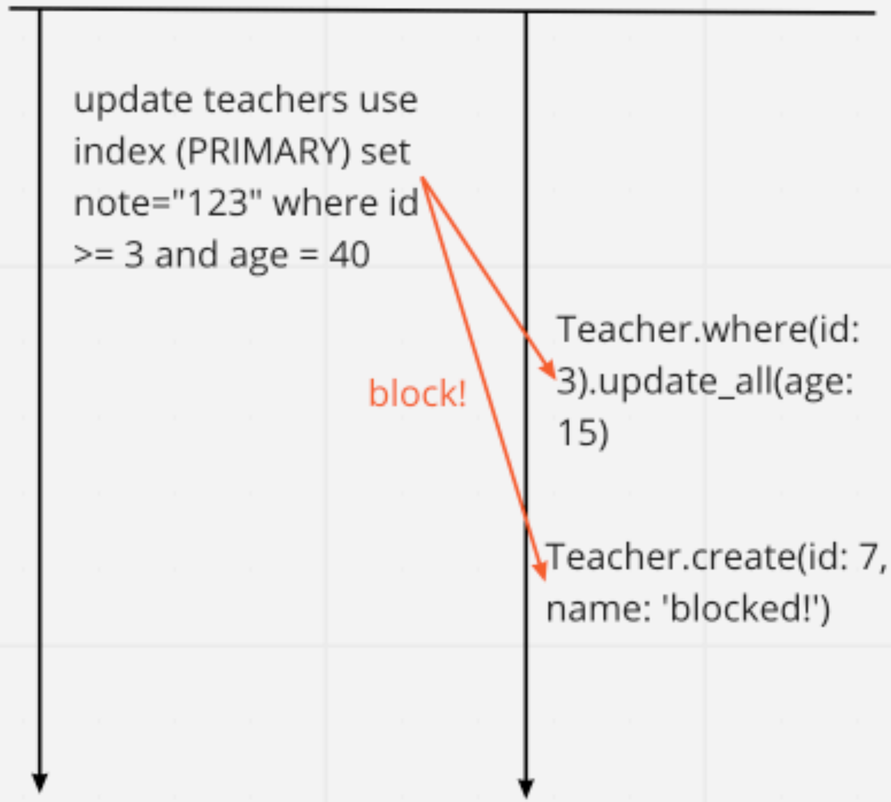
- Lock Mode
  - Share (S) / Exclusive (X)
- Lock Type
  - Record Lock (Table / Row)
  - Intention Lock (Table Level)
  - Auto-Incr Lock
  - Gap Lock (Prevent Insert)
  - Insert Intention Lock
  - Next-Key Lock



MySQL Lock

```
LATEST DETECTED DEADLOCK
------------------------
2022-04-22 10:36:54 277165582080
*** (1) TRANSACTION:
TRANSACTION 1872, ACTIVE 3 sec inserting
mysql tables in use 1, locked 1
LOCK WAIT 3 lock struct(s), heap size 1128, 2 row lock(s)
MySQL thread id 25, OS thread handle 278780622592, query id 203
172.17.0.1 root update
INSERT INTO `teachers` (`id`) VALUES (6)

*** (1) HOLDS THE LOCK(S):
RECORD LOCKS space id 3 page no 4 n bits 72 index PRIMARY of table
`test`.`teachers` trx id 1872 lock_mode X
Record lock, heap no 1 PHYSICAL RECORD: n_fields 1; compact
format; info bits 0
 0: len 8; hex 73757072656d756d; asc supremum;;


*** (1) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 3 page no 4 n bits 72 index PRIMARY of table
`test`.`teachers` trx id 1872 lock_mode X insert intention waiting
Record lock, heap no 1 PHYSICAL RECORD: n_fields 1; compact
format; info bits 0
 0: len 8; hex 73757072656d756d; asc supremum;;

*** (2) TRANSACTION:
TRANSACTION 1873, ACTIVE 3 sec inserting
mysql tables in use 1, locked 1
LOCK WAIT 3 lock struct(s), heap size 1128, 2 row lock(s)
MySQL thread id 26, OS thread handle 278783792896, query id 204
172.17.0.1 root update
INSERT INTO `teachers` (`id`) VALUES (10)

*** (2) HOLDS THE LOCK(S):
RECORD LOCKS space id 3 page no 4 n bits 72 index PRIMARY of
table `test`.`teachers` trx id 1873 lock_mode X
Record lock, heap no 1 PHYSICAL RECORD: n_fields 1; compact
format; info bits 0
 0: len 8; hex 73757072656d756d; asc supremum;;


*** (2) WAITING FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 3 page no 4 n bits 72 index PRIMARY of
table `test`.`teachers` trx id 1873 lock_mode X insert intention
waiting
Record lock, heap no 1 PHYSICAL RECORD: n_fields 1; compact
format; info bits 0
 0: len 8; hex 73757072656d756d; asc supremum;;

*** WE ROLL BACK TRANSACTION (2)
------------
```

Range Update



New table

id	name	age	note
2	a	10	
3	b	12	
4	c	13	
100	d	15	

- id: primary key (clustered index)
- name: unique secondary key
- age: non unique secondary key

Range Update

- Lock every rows even it is not satisfied where condition

"When using the default `REPEATABLE READ` isolation level, the first `UPDATE` acquires an x-lock on each row that it reads and does not release any of them:"

- https://dev.mysql.com/doc/refman/8.0/en/innodb-transaction-isolation-levels.html#isolevel_repeatable-read

MySQL Lock - Clustered Index

- Hit => Row Lock
- Not Hit => Next Key Lock
- Range Search => Lock every row it read

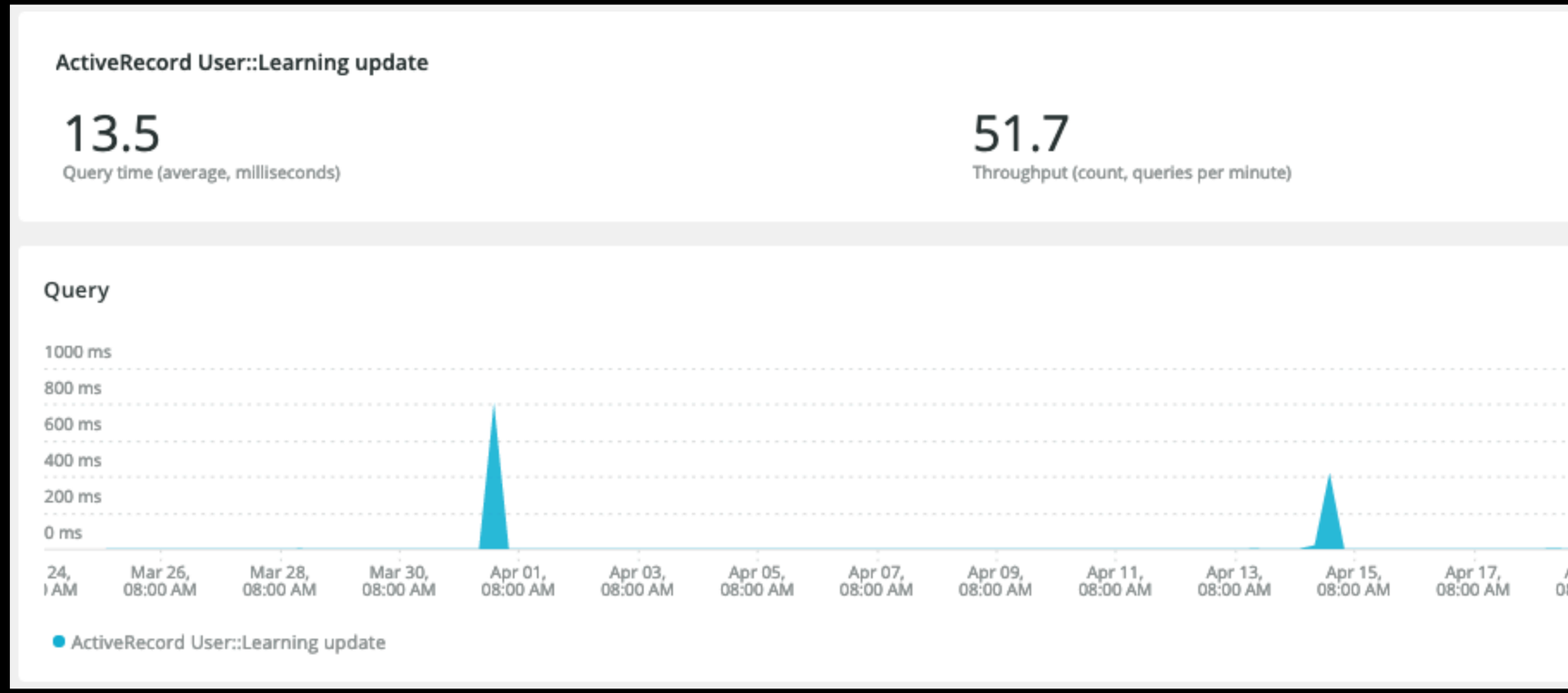
Real Case 1

- Hourly::CheckAppointmentRequestExpired
  - 預約 Appointment 時，會需要老師 confirm，如果超過時間沒有 confirm 則更新 Appointment 狀態為 expired
- Metrics: Appointment Update
- Metrics: Appointment Create

```
# Query_time: 15.574314  Lock_time: 0.000079 Rows_sent: 0  Rows_examined: 7484706
SET timestamp=1651940578;
UPDATE `appointments` SET `appointments`.`state` = 10, `appointments`.`state_change_time` =
'2022-05-07 16:22:43' WHERE `appointments`.`deleted_at` IS NULL AND ((state = 0) and
(confirm_expired_time <= '2022-05-07 16:20:02.597431'));
```

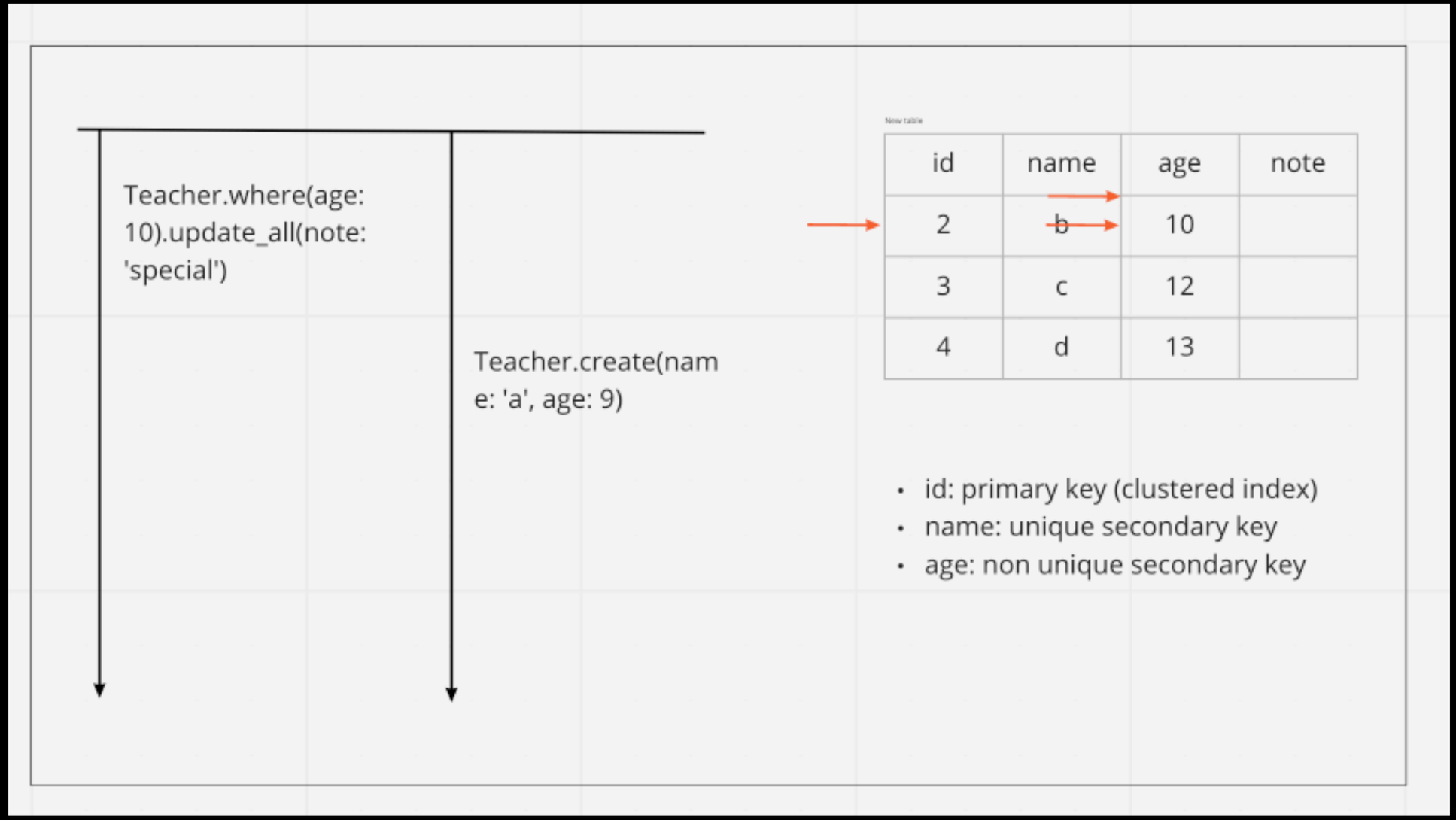
Real Case 2

- Monthly::DividedRatio
- metrics reference




```
User::Learning.update_all('intrested_ratio = intrested_ratio / 2')
Locale::LanguageTagSearch.update_all('count = count / 2')
```

Not Unique Secondary Index



Secondary Index

- Type
  - Unique => Almost Like Clustered Index
  - Not Unique => require gap lock even hit
- Keep primary key reference

> “For other search conditions, and for non-unique indexes, InnoDB locks the index range scanned, using gap locks or next-key locks to block insertions by other sessions into the gaps covered by the range.”
>
> - https://dev.mysql.com/doc/refman/8.0/en/innodb-locks-set.html

How To Solve

- 批次更新 (find_in_batch)
- 篩選出 clustered id 再更新，避免多餘的 gap lock

```
Ids = Collection.where(conditions).pluck(:id)

Collection.where(id: ids).update_all(….)
```

- 改成 read-committed
- 檢查 ORM 產生的 SQL

Why not set Read-Committed as default?

- Controversial

"In general I think good practice is to use READ COMITTED isolation mode as default and change to REPEATABLE READ for those applications or transactions which require it."

- https://www.percona.com/blog/2015/01/14/mysql-performance-implications-of-innodb-isolation-modes/

"THE GENERAL RULE could be the following :

- if your queries and transactions are short : use rather the default REPEATABLE-READ mode!
- if your queries are long and reading a lot of data which are likely to be modified by other transactions in parallel : then use the READ-COMMITTED mode

"

- https://dev.mysql.com/blog-archive/performance-impact-of-innodb-transaction-isolation-modes-in-mysql-5-7/