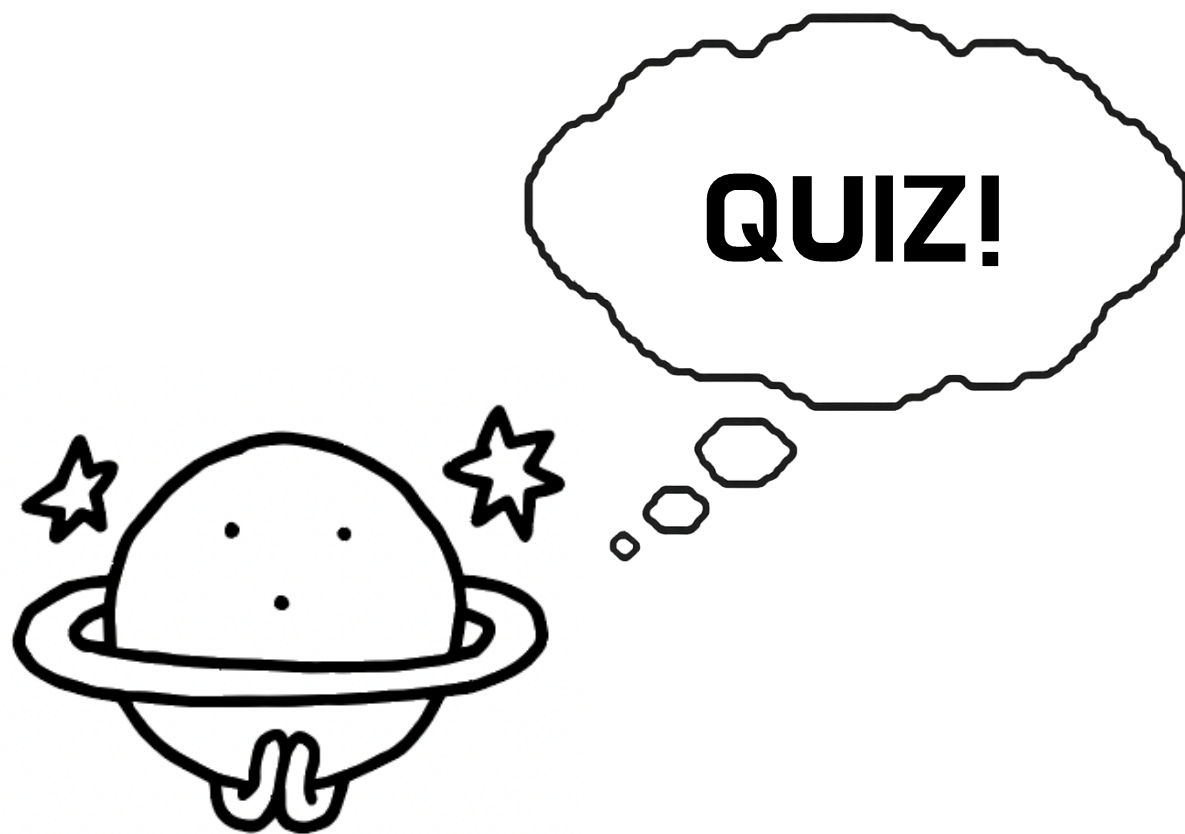


MySQL ?????????? ?????

우아한테크코스 BE 7기 메이





| member

id	long
name	<u>varchar(255)</u>
email	varchar(255)

Row: 100,000



맞춰보세요

1번 쿼리

```
ALTER TABLE member  
ADD COLUMN age INT DEFAULT 25;
```

Query OK, _ rows affected;

2번 쿼리

```
UPDATE member SET age = 25;
```

Query OK, _ rows affected;

3번 쿼리

```
UPDATE member SET age = 100;
```

Query OK, _ rows affected;

MySQL 8.0.29

| member

id	long
name	<u>varchar(255)</u>
email	<u>varchar(255)</u>
age	int

Row: 100,000



1번 쿼리

```
ALTER TABLE member  
ADD COLUMN age INT DEFAULT 25;
```

```
mysql> ALTER TABLE member ADD COLUMN age INT DEFAULT 25;  
Query OK, 0 rows affected (0.06 sec)
```

2번 쿼리

```
UPDATE member SET age = 25;
```

```
mysql> UPDATE member SET age = 25;  
Query OK, 0 rows affected (0.06 sec)
```

3번 쿼리

```
UPDATE member SET age = 100;
```

```
mysql> UPDATE member SET age = 50;  
Query OK, 100000 rows affected (0.66 sec)
```



```
-- expand) role 필드 추가
ALTER TABLE account ADD COLUMN role VARCHAR(255) NOT NULL DEFAULT 'USER';

UPDATE account SET role = 'USER'; -- claude code가 추가하라고 함
```



coderabbitai bot [2 weeks ago](#)

[@seaniiio](#), MySQL 8.0의 Instant DDL 동작 방식 때문입니다.

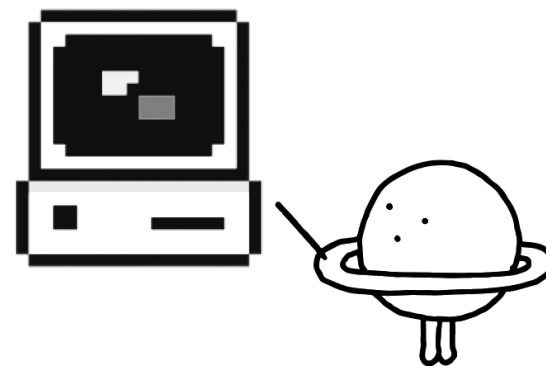
MySQL 8.0의 Instant DDL 동작

MySQL 8.0에서 `ALTER TABLE ADD COLUMN ... DEFAULT 'value'` 를 실행하면:

- 컬럼 정의와 기본값이 테이블 메타데이터에만 저장됩니다
- 기존 행들에는 물리적으로 값이 쓰여지지 않습니다
- 새로 INSERT되는 행들만 물리적으로 'USER' 값을 가집니다

음..어떻게..?





MySQL Instant 알고리즘

우아한테크코스 BE 7기 메이



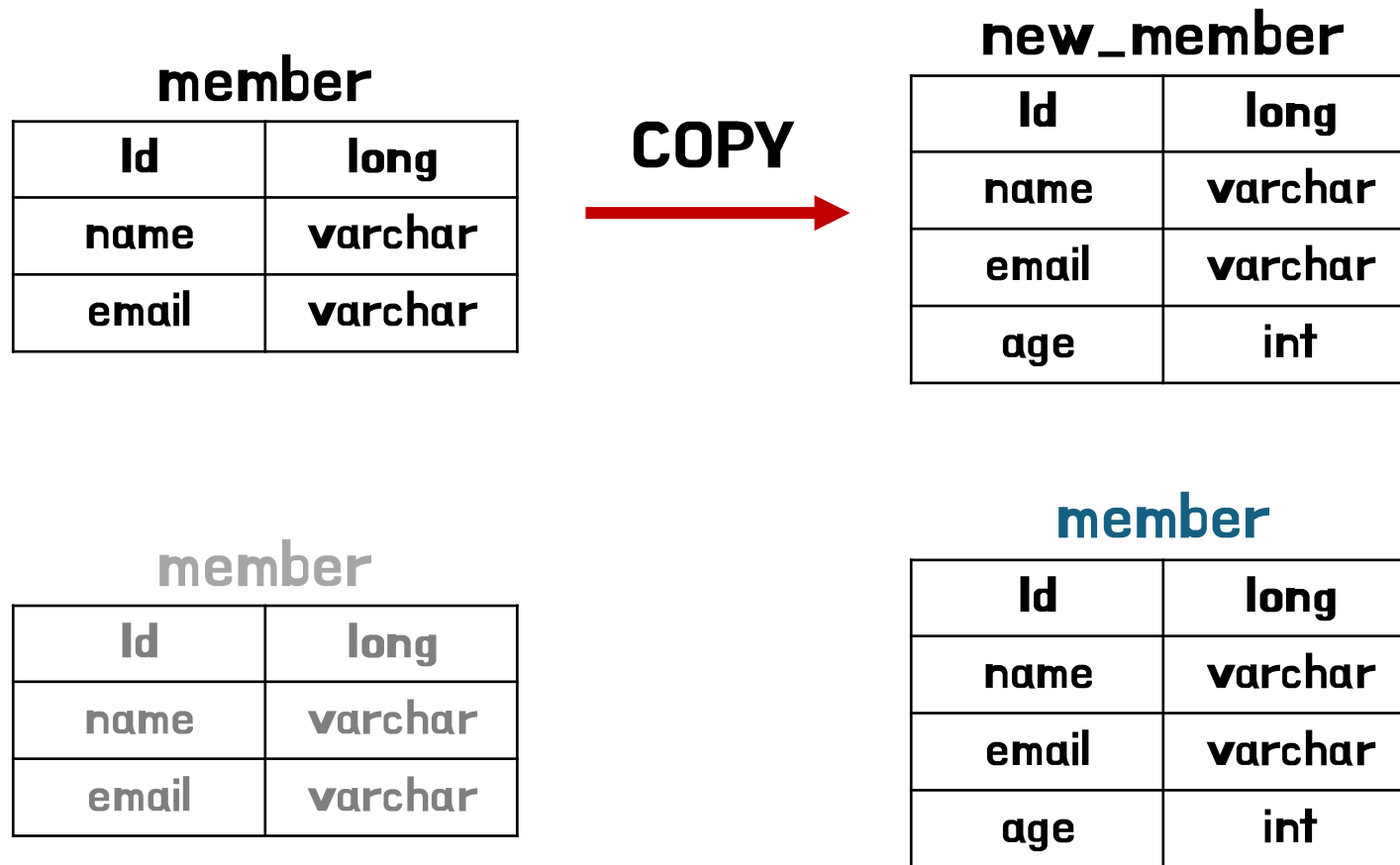
MySQL DDL 알고리즘





MySQL DDL 알고리즘 - COPY

ALTER TABLE member ADD COLUMN age int DEFAULT 25, **ALGORITHM = COPY;**





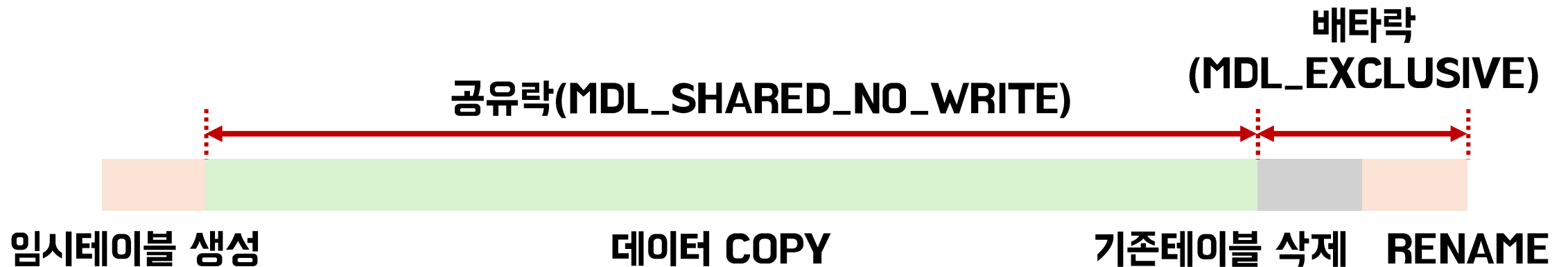
MySQL DDL 알고리즘 - COPY

member	
Id	long
name	varchar
email	varchar

COPY



new_member	
Id	long
name	varchar
email	varchar
age	int





MySQL DDL 알고리즘 - INPLACE

- MySQL 5.6 도입
- 테이블 직접 수정
- Online DDL

ALTER TABLE member ADD COLUMN age int DEFAULT 25, **ALGORITHM = INPLACE;**

member

Id	long
name	varchar
email	varchar
age	int

배타락
(MDL_EXCLUSIVE)

배타락
(MDL_EXCLUSIVE)



준비

DDL 실행



마무리



MySQL DDL 알고리즘 - INPLACE

ALTER TABLE member **ADD INDEX** idx_name(name), **ALGORITHM = INPLACE;**

준비

DDL 실행

세컨더리 인덱스 생성중,,



1시간 후..



마무리

실패했으니 롤백합니다~

준비

DDL 실행

세컨더리 인덱스 생성중,,



ERROR 1799: Creating index 'idx_name' required more than 'innodb_online_alter_log_max_size' bytes of modification log.



Online DDL 로그 버퍼 꽉 차서 실패



MySQL DDL 알고리즘 - INSTANT

- MySQL 8.0.12 도입
- 실제 row 데이터를 수정하지 않고, **메타데이터만 수정**

ALTER TABLE member ADD COLUMN age int DEFAULT 25, **ALGORITHM = INSTANT;**





MySQL DDL 알고리즘 - INSTANT

Instant V1

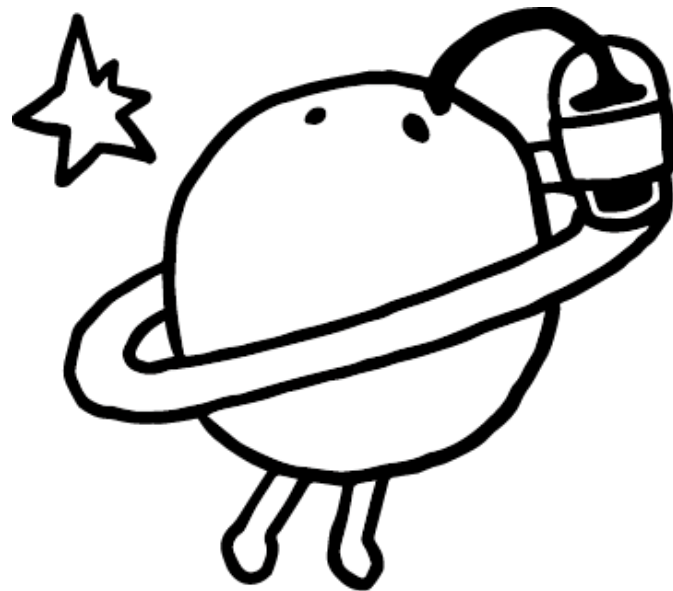
- MySQL 8.0.12
- 마지막 위치에만 컬럼 추가 가능
- 컬럼 삭제 불가능

Instant V2

- MySQL 8.0.29
- **row_version** 개념 추가
- 위치에 상관 없이 컬럼 추가 가능
- 컬럼 삭제 가능

INSTANT 알고리즘

동작 원리 ☆



예시 상황

ALTER TABLE member ADD COLUMN age int DEFAULT 25, **ALGORITHM = INSTANT;**
ALTER TABLE member DROP COLUMN email, **ALGORITHM = INSTANT;**

| member

id	long
name	<u>varchar(255)</u>
email	varchar(255)



| member

id	long
name	<u>varchar(255)</u>
email	<u>varchar(255)</u>
age	int

[id][name][email]
[1][“may”][“may@example.com”]
[2][“sw”][“sw@example.com”]

Version	스키마	이벤트
0	id, name, email	원본 테이블
1	id, name, email, age	age 컬럼 추가 (version_added=1)
2	id, name, age	email 컬럼 drop (version_dropped=2)



v1 테이블 메타데이터

ALTER TABLE member ADD COLUMN age int DEFAULT 25, **ALGORITHM = INSTANT;**

member	
id	long
name	<u>varchar(255)</u>
email	<u>varchar(255)</u>
age	int

total_row_version = 1



```
{  
  "name": "age",  
  "ordinal_position": 4,  
  "se_private_data": "default=80000019;physical_pos=5;table_id=2657 version_added=1;"  
}
```

member.ibd



v2 테이블 메타데이터

ALTER TABLE member DROP COLUMN email, **ALGORITHM = INSTANT;**

| member

id	long
name	varchar(255)
email	varchar(255)
age	int

total_row_version = 2



```
{  
  "name": "!hidden_dropped_v2_p4_email",  
  "hidden": 2,  
  "ordinal_position": 6,  
  "se_private_data": "physical_pos=4 version_dropped=2;"  
}
```

member.ibd

	[id][name][email][age]
v0에서 추가	[1][“may”][“may@example.com”] [2][“sw”][“sw@example.com”]
v1에서 추가	[3][“cool”][“cool@example.com”][100]
v2에서 추가	[4][“sean”][25]



실제 데이터 확인하기

member.ibd

email 여전히 존재

00010090	00	00	af	01	10	6d	61	79	6d	61	79	40	65	78	61	6dmaymay@exam	v0
000100a0	70	6c	65	2e	63	6f	6d	0e	02	00	00	00	18	00	2e	80	ple.com.....	
000100b0	00	00	00	00	00	00	02	00	00	00	00	65	08	82	00	00e....	
000100c0	00	ad	01	10	73	77	73	77	40	65	78	61	6d	70	6c	65swsw@example	v0
000100d0	2e	63	6f	6d	10	04	00	01	40	00	20	00	35	80	00	00	.com....@. .5...	
000100e0	00	00	00	00	03	00	00	00	00	65	17	81	00	00	00	b4e.....	
000100f0	01	10	63	6f	6f	6c	63	6f	6f	6c	40	65	78	61	6d	70	..coolcool@examp	v1
00010100	6c	65	2e	63	6f	6d	80	00	00	64	04	00	02	40	00	28	le.com...d...@.(
00010110	ff	5e	80	00	00	00	00	00	00	04	00	00	00	00	65	23	.^.....e#	
00010120	82	00	00	00	b6	01	10	79	65	61	6e	80	00	00	19	00sean.....	v2

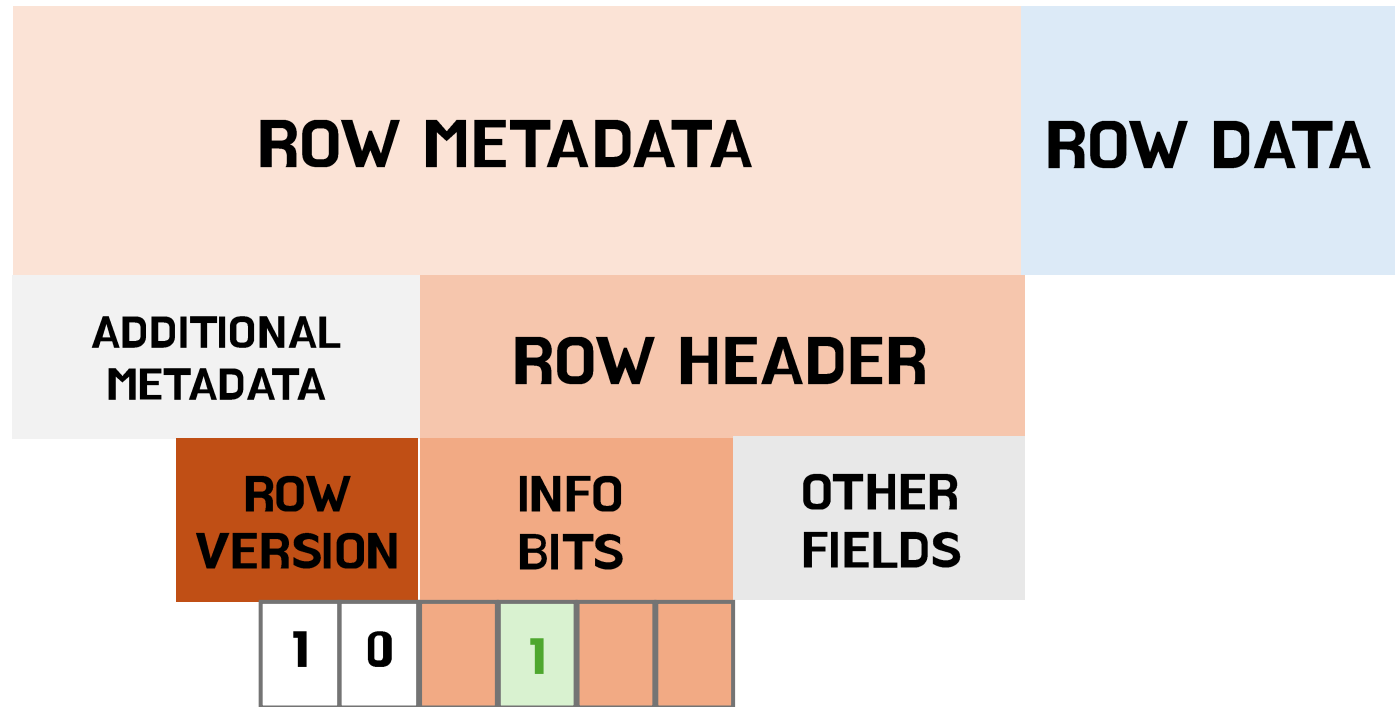
age=100

age=25

(부호 있는 INT는 0x80000000 더해서 저장)

InnoDB row 구조

[id][name][age]
[1][“May”][25]

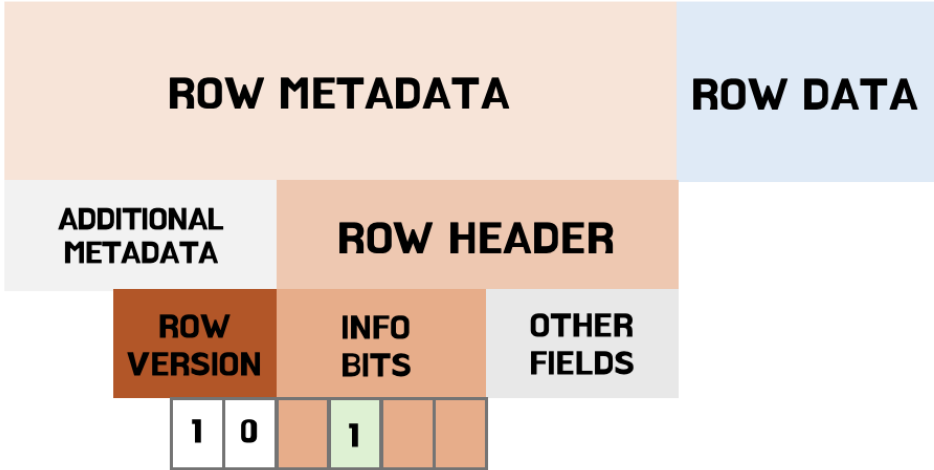


VERSION_FLAG

테이블의 row_version이 2일 때 추가된 row



실제 데이터 확인하기



00010090	00	00	af	01	10	6d	61	79	6d	61	79	40	65	78	61	6dmaymay@exam
000100a0	70	6c	65	2e	63	6f	6d	0e	02	00	00	00	18	00	2e	80	ple.com.....
000100b0	00	00	00	00	00	00	02	00	00	00	0000	5	08	82	00	00e....
000100c0	00	ad	01	10	73	77	73	77	40	65	78	61	6d	70	6c	65swsw@example
000100d0	2e	63	6f	6d	10	04	00	01	40	00	20	00	35	80	00	00	Id=3com....@. .5...
000100e0	00	00	00	00	03	00	00	0100	5	17	81	00	00	00	00	b4e.....
000100f0	01	10	63	6f	6f	6c	63	6f	6f	6c	40	65	78	61	6d	70	..coolcool@examp
00010100	6c	65	2e	63	6f	6d	80	00	00	64	04	00	02	40	00	28	le.com...d...@.(
00010110	ff	5e	80	00	00	00	00	00	00	04	00	00	00	00	00	00	.^.....e#
00010120	82	00	00	00	b6	01	10	73	65	61	6e	80	00	00	19	00sean.....

v0

v1

v2

member.columns

column	version_added	version_dropped	default_value
id	0	0	
name	0	0	
email	0	2	
age	1	0	25



데이터 Fetch – version_added, version_dropped가 0

member.columns

column	version_added	version_dropped	default_value
id	0	0	
name	0	0	
email	0	2	
age	1	0	25

member.rows

row_version	id	name	email	age
0	1	may	may@example.com	-
0	2	sw	sw@example.com	-
1	3	cool	cool@example.com	100
2	4	sean	-	25



데이터 Fetch – version_dropped가 1 이상

member.columns

column	version_added	version_dropped	default_value
id	0	0	
name	0	0	
email	0	2	
age	1	0	25

member.rows

row_version	id	name	email	age
0	1	may	may@example.com	-
0	2	sw	sw@example.com	-
1	3	cool	cool@example.com	100
2	4	sean	-	25



데이터 Fetch – version_added가 1 이상

member.columns

column	version_added	version_dropped	default_value
id	0	0	
name	0	0	
email	0	2	
age	1	0	25

member.rows

row_version	id	name	email	age
0	1	may	may@example.com	25
0	2	sw	sw@example.com	25
1	3	cool	cool@example.com	100
2	4	sean	-	25

default_value 읽음

실제 데이터 읽음

데이터 Fetch 결과

SELECT * FROM member;



id	name	age
1	may	25
2	sw	25
3	cool	100
4	sean	25

INSTANT 알고리즘 사용 시 주의할 점



ROW_VERSION LIMIT

When a table with instantly added or dropped columns is rebuilt by table-rebuilding ALTER TABLE or OPTIMIZE TABLE operation, the `TOTAL_ROW_VERSIONS` value is reset to 0. The maximum number of row versions permitted is 64 (255 as of MySQL 9.1.0), as each row version requires additional space for table metadata. When the row version limit is reached, `ADD COLUMN` and `DROP COLUMN` operations using `ALGORITHM=INSTANT` are rejected with an error message that recommends rebuilding the table using the `COPY` or `INPLACE` algorithm.

**row_version이 최대 64
(MySQL 9.1.0 이후 최대 255)**

ROW_VERSION LIMIT

(INSTANT 알고리즘 적용되는 DDL)

ALTER TABLE member ADD COLUMN age int DEFAULT 25;

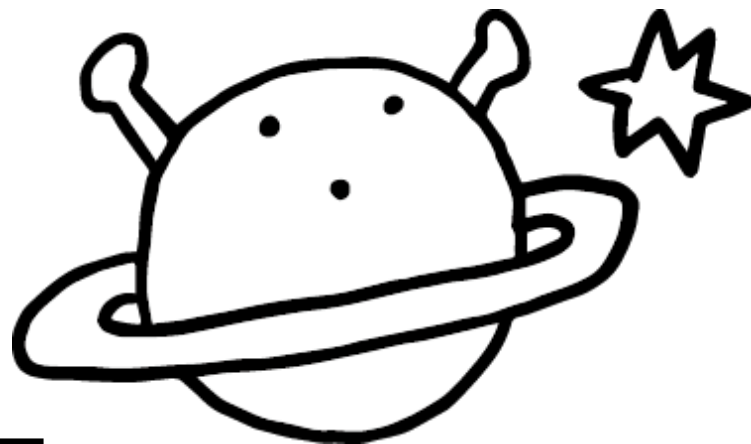


ERROR 4092 (HY000): Maximum row versions reached for table instant_test/member.
No more columns can be added or dropped instantly. Please use COPY/INPLACE.

해결 방법

- 1) 다른 알고리즘(COPY, INPLACE) 사용하도록 명시
- 2) 테이블 리빌드를 통해 테이블의 row_version 초기화

- **SELECT 쿼리의 성능 저하 예상**
- **물리적 백업 도구 사용 시 INSTANT DDL을 지원하는지 확인**
- **FTS 인덱스 걸린 테이블에는 사용 불가**



결론

- 지금까지 잘 사용해온 **INSTANT** 알고리즘을 이해
- 가이드라인을 통해 내가 사용하는 **DDL**이 어떤 알고리즘에서 동작하는지 알 수 있다.

Table 17.17 Online DDL Support for Column Operations

Operation	Instant	In Place	Rebuilds Table	Permits Concurrent DML	Only Modifies Metadata
Adding a column	Yes*	Yes	No*	Yes*	Yes
Dropping a column	Yes*	Yes	Yes	Yes	Yes
Renaming a column	Yes*	Yes	No	Yes*	Yes
Reordering columns	No	Yes	Yes	Yes	No
Setting a column default value	Yes	Yes	No	Yes	Yes
Changing the column data type	No	No	Yes	No	No
Extending VARCHAR column size	No	Yes	No	Yes	Yes
Dropping the column default value	Yes	Yes	No	Yes	Yes
Changing the auto-increment value	No	Yes	No	Yes	No*
Making a column NULL	No	Yes	Yes*	Yes	No
Making a column NOT NULL	No	Yes*	Yes*	Yes	No
Modifying the definition of an ENUM or SET column	Yes	Yes	No	Yes	Yes

Q&A

