#ASLI ENGINEERING

Github Outage Dissection: When DB hits integer limit

SWIPE

BY

ARPIT BHAYANI

Dissecting GitHub's Outage

When Database ID hits the maximum value

On May 5, 2020, one of the tables with AUTO_INCREMENT ID column reachead the maximum value.

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(8) NOT NULL
);
```

id	name

INT on MySQL is 4 bytes & has range (signed)

$-2,147,483,648$ to $2,147,483,647$ ← GitHub reached this value on its ID coulum for Installation Token

-2^{31} to $2^{31}-1$

Impact: Actions, Pages, Dependabot

When the ID reached the max value

id	.....
2147483647	

Inserts started failing !! (as per GitHub)

Errors thrown : Range Error

Note: When I tried Duplicate Key Error

ARPIT BHAYANI

What does this mean ?

Auto_incrementing stops when max value reached

1, 2, 3, 4, 5, 6, ..., 100, ..., 2147483647, 2147483647, 2147483647...

Keep getting the same number

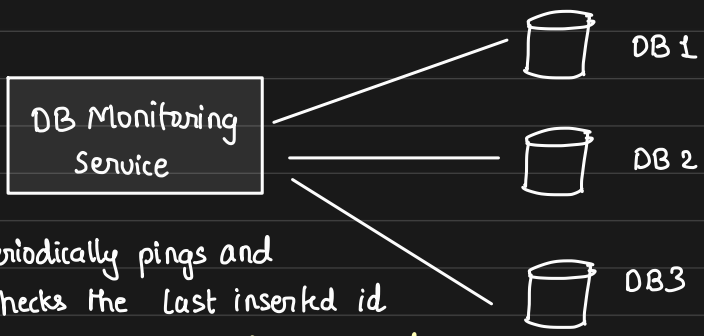
For MySQL: 2147483647 + 1 = 2147483647

So, it is actually trying to insert the row with the same ID again and again → **Duplicate Key Error**

To be honest, a very poor error
Should have been a special error considering how critical the error is.

How did GitHub fixed the issue? No mention

Preventive Measure



Periodically pings and checks the last inserted id and alerts when 70% reached

ARPIT BHAYANI

How to mitigate?

Approach 1: Make ID UNSIGNED / BIGINT
Approach 2: Swap the table

Approach 1: Make ID UNSIGNED

When table is small & you do not use negative IDs

Range: SIGNED: -2^{31} to $2^{31}-1$ ↓ Double the space
UNSIGNED: 0 to $2^{32}-1$

Steps:

Do not lock rows during alteration ↓

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

SET FOREIGN_KEY_CHECKS = 0;   ← Do not checke for F.Key to do it quick

ALTER TABLE users
  CHANGE id id INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
ALGORITHM = COPY, LOCK = SHARED
```

↑ Copy the table with the new altered column and not do it "IN_PLACE"

ARPIT BHAYANI

Approach 2: Swap the table

Idea: Create another table (like the original table but empty) with larger ID range

Note: This solution would work when you can live without old data

Steps:

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
```
← Do not lock rows during alteration

```
SET FOREIGN_KEY_CHECKS = 0;
```
← Do not checke for F.Key to do it quick

```
CREATE TABLE users_2 LIKE users;
```
← Create empty table

```
ALTER TABLE users_2
  CHANGE id id BIGINT(12) UNSIGNED NOT NULL AUTO_INCREMENT;
```
← Make ID larger

```
ALTER TABLE users_2 AUTO_INCREMENT = 2147483648;
```
← set auto_inc to the next valu

```
RENAME TABLE users to users_old, users_2 to users;
```
← Rename table

```
INSERT INTO users SELECT * FROM users_old;
```
↑ Copy old data back

ARPIT BHAYANI