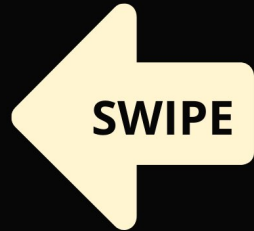




**#ASLI ENGINEERING**

# Github Outage Dissection: When DB hits integer limit



**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 reached the maximum value.

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

→ signed

INT on MySQL is 4 bytes & has range

-2,147,483,648 to 2,147,483,647  
-2<sup>31</sup> to 2<sup>31</sup> - 1

← GitHub reached this value on its ID column for Installation Token

Impact: Actions, Pages, Dependabot

When the ID reached the max value

id	.....
_____	_____
_____	_____
_____	_____
_____	_____
2147483647	_____

Inserts started failing !!

Errors thrown: Range Error

as per GitHub

Note: When I tried Duplicate Key Error

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:  $2147483647 + 1 = 2147483647$

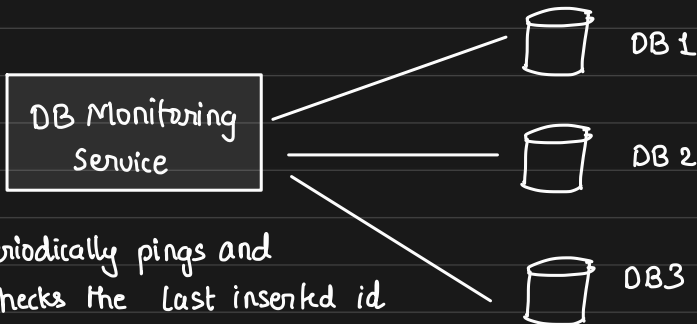
So, it is actually trying to insert the row with the same ID again and again

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

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 check 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"

## Approach 2: Swap the table

like the original table but empty

Idea: Create another table with larger ID range

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

Steps:

Do not lock rows during alteration

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

SET FOREIGN\_KEY\_CHECKS = 0;

Do not check for F-Key  
to do it quick

CREATE TABLE users\_2 LIKE users; ← create empty table

ALTER TABLE users\_2

Make ID larger

CHANGE id id BIGINT(12) UNSIGNED NOT NULL AUTO-INCREMENT;

ALTER TABLE users\_2 AUTO-INCREMENT = 2147483648; ← set auto-inc  
to the next value

RENAME TABLE users to users\_old, users\_2 to users; ← Rename table

INSERT INTO users SELECT \* FROM users\_old;

↑  
Copy old data back