

Waiting for table flush

Der Status `Waiting for table flush` entsteht meistens durch ein klassisches **Henne-Ei-Problem** im MySQL-Query-Locking: Ein Prozess fordert das Schließen/Leeren der Tabellencaches an (z. B. ein Backup, `FLUSH TABLES`, `ALTER TABLE` oder `ANALYZE TABLE`), wird aber selbst von einer lang laufenden `SELECT`-Query blockiert. Alle *darauffolgenden* Abfragen auf diese Tabelle müssen dann im Status `Waiting for table flush` warten.

Hier ist die Schritt-für-Schritt-Anleitung zur Ursachenforschung und Prävention:

1. Die genaue Ursache ermitteln

Um den "Übeltäter" (den Blocker) zu finden, musst du die Kette der abhängigen Abfragen analysieren.

Schritt A: Langläufer identifizieren (`SHOW FULL PROCESSLIST`)

Führe folgenden Befehl aus:

SQL

```
SELECT ID, USER, HOST, DB, COMMAND, TIME, STATE, INFO
FROM information_schema.PROCESSLIST
WHERE COMMAND != 'Sleep'
ORDER BY TIME DESC;
```

Wonach du suchen musst:

1. **Der Flush-Auslöser:** Suche nach Abfragen mit `FLUSH TABLES`, `ALTER TABLE`, `RENAME TABLE`, `OPTIMIZE TABLE` oder `ANALYZE TABLE`. Diese haben oft eine moderate `TIME`.
2. **Der eigentliche Blocker:** Suche nach **sehr alten** `SELECT`-Abfragen (hohe `TIME`), die **vor** dem Flush gestartet wurden. Das sind häufig schlechte Queries ohne passenden Index oder ungeschlossene Transaktionen.

Schritt B: Exakte Sperren über das Performance Schema auslesen

Falls MySQL 5.7/8.0+ im Einsatz ist, zeigt dir das `performance_schema` genau, welcher Prozess auf welche Metadata Lock (MDL) wartet:

SQL

```
SELECT
    waiting_p.ID AS waiting_thread_id,
    waiting_p.INFO AS waiting_query,
    blocking_p.ID AS blocking_thread_id,
    blocking_p.INFO AS blocking_query
FROM performance_schema.metadata_locks waiting_ml
JOIN information_schema.PROCESSLIST waiting_p
    ON waiting_ml.OWNER_THREAD_ID = waiting_p.ID
JOIN performance_schema.metadata_locks blocking_ml
    ON waiting_ml.OBJECT_SCHEMA = blocking_ml.OBJECT_SCHEMA
    AND waiting_ml.OBJECT_NAME = blocking_ml.OBJECT_NAME
JOIN information_schema.PROCESSLIST blocking_p
    ON blocking_ml.OWNER_THREAD_ID = blocking_p.ID
WHERE waiting_p.STATE = 'Waiting for table flush'
    AND blocking_p.ID != waiting_p.ID;
```

2. Typische Ursachen & wie man sie behebt

Ursache 1: Automatische Backups (`mysqldump` / `mariadb-dump`)

Klassischer Auslöser: Ein Backup-Job führt `FLUSH TABLES WITH READ LOCK` aus. Wenn parallel eine lange `SELECT`-Abfrage läuft, blockiert das Backup alle schreibenden und lesenden Zugriffe.

- **Lösung (InnoDB):** Verwende bei `mysqldump` immer das Flag `--single-transaction`. Dadurch wird die Tabelle nicht mehr global gesperrt:

Bash

```
mysqldump --single-transaction --quick -u root -p meinedatenbank > backup.sql
```

Ursache 2: Langsame `SELECT`-Queries ohne Index

Ein automatischer Wartungsjob (z. B. `ANALYZE TABLE`) oder DDL-Befehl möchte ein Table-Flush ausführen. Ein schlechter `SELECT`, der 5 Minuten für einen Full-Table-Scan braucht, blockiert den Flush.

- **Lösung:**

- Analysiere langsame Abfragen mit `EXPLAIN` und setze fehlende Indizes.
- Begrenze die maximale Laufzeit von Leseabfragen über Timeouts (z. B. in MySQL 8.0+):

SQL

```
SET GLOBAL max_execution_time = 30000; -- bricht SELECTs nach 30 Sekunden ab
```

Ursache 3: Nicht committete Transaktionen

Ein Entwickler oder eine Anwendung hat `BEGIN` ausgeführt, ein `SELECT` gemacht und vergessen, `COMMIT` oder `ROLLBACK` aufzurufen. Der Thread schläft (`Sleep`), hält aber die Metadata-Lock.

- **Lösung:** Aktive Transaktionen identifizieren:

SQL

```
SELECT * FROM information_schema.innodb_trx;
```

Findest du dort alte Transaktionen (`trx_started`), kannst du diese gezielt mit `KILL [trx_mysql_thread_id]` beenden.

Ursache 4: Zu kleine Table Caches

Wenn MySQL ständig Tabellen schließen muss, weil der Cache voll ist, steigt die Wahrscheinlichkeit für Flush-Konflikte.

- **Lösung:** Überprüfe in der `my.cnf` / `my.ini` folgende Variablen und erhöhe sie gegebenenfalls:

Ini, TOML

```
table_open_cache = 4096
table_definition_cache = 4096
```

Zusammenfassung zur Prävention

Problem	Schnelle Abhilfe	Langfristige Prävention
Backups	<code>--single-transaction</code> nutzen	Keine <code>FLUSH TABLES WITH READ LOCK</code> auf InnoDB nutzen
Langläufer	Prozess mit <code>KILL <ID></code> beenden	Indizes optimieren & Query-Timeouts setzen

Problem	Schnelle Abhilfe	Langfristige Prävention
Offene Transaktionen	Idle-Connections kappen	<code>interactive_timeout</code> & <code>wait_timeout</code> reduzieren

Created 2026-07-23 09:36:29 UTC by Mich3l
Updated 2026-07-23 09:50:34 UTC by Mich3l