TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Informatik
Professur Datenverwaltungssysteme

Advanced Management of Data
Exercise 6 Topic 2:

Extensions of SQL

Triggers
Academic example

- now, we have two tables for logging, what's going on with numbers, but one could easily mess up with them and `UPDATE`, `DELETE` or `TRUNCATE` data from them, so we don't know what happened
- Task: use triggers to prevent this, so that one can only insert something to `numbers_log` and `numbers_log_query`
- disclaimer: of course this won't be an elegant solution but merely a workaround, as one wouldn't grant the database user any privileges, that it shouldn't have and simply revoke those privileges, but consider this just another academic example like the rest of this exercise

Triggers
Exceptional

- row-level trigger returning `NULL` prevent further execution, but triggers on `TRUNCATE` are only allowed as statement-level trigger and those should always return `NULL` and don't prevent the execution
- therefore we have to raise an exception in before to prevent further execution

SAMPLE CODE GOT PROVIDED DURING THE EXERCISE

Triggers
Distinct view

- inserting the same values again and again to our `numbers` table to test some triggers might get boring, so let's change our view by creating a new `VIEW` that only shows distinct values and does some sorting

```
CREATE OR REPLACE VIEW numbers_distinct AS SELECT DISTINCT number FROM numbers
ORDER BY number;
```

- unfortunately, this view isn't automatically updatable as it contains a `DISTINCT` clause at the top level, so the system doesn't allow `INSERT`, `UPDATE` and `DELETE` statements on the `VIEW`, as it can't translate them to corresponding statements on the base relation
- therefore we have to solve this
- Task: use triggers to enable the usage of `INSERT`, `UPDATE` and `DELETE` on `numbers_distinct`

Triggers
Update View

SAMPLE CODE GOT PROVIDED DURING THE EXERCISE

Triggers
Working View

- by using TG_OP to get the type of operation that should be performed, we can do everything with just one trigger function, but it is also possible to use one function for each case
- of course we would need to write more code with different trigger functions and we would have to write even more code, as we would need different triggers and now we can define just one trigger for all cases

```
CREATE TRIGGER numbers_distinct_change INSTEAD OF INSERT OR UPDATE OR
DELETE ON numbers_distinct
  FOR EACH ROW
    EXECUTE PROCEDURE numbers_distinct_update();
```

- now you can try to change numbers_distinct and see how numbers is modified