Click icon to add picture

Database Lab
User-defined Functions and Stored Procedures

Fall Term 2023
Dr. Andreas Geppert
geppert@acm.org

Topics

- conceptual design
- logical design
- consistency constraints
- data manipulation
- queries
- transactions
- views
- **stored procedures and user-defined functions**
- triggers
- security

Stored Procedures and User-defined Functions

- User-defined routines:
  - Functions
  - Procedures
  - Trigger
- SQL with control flow operators
  - implementation languages SQL, PL/pgSQL, ...
- scalar functions
  - Functions always returning a single value
- Table functions
  - functions that return tables („parameterized views“)
- Stored procedures
  - „Functions“ that (can) contain updates
- Trigger / rules
  - actions which are executed event-based

Stored Procedures and User-defined Functions

- Parameterless scalar function:
- create function countCars() returns integer as $$
  select count(*)::integer from vehicle $$ language SQL;

  select countCars();

- Scalar function with parameters:
- create function countCars(loc integer) returns integer as $$
  select count(*)::integer from vehicle where home = loc $$ language SQL;

  select name, countCars(id) * 100 / countCars() as percent
    from location;

Stored Procedures and User-defined Functions

- parameterized table function

```
create function getVehiclesByMake(m Text)
    returns table(licensePlate Text, model Text) as
    $$ select licensePlate, model from vehicle where make = m
    $$ language SQL;

select * from getVehiclesByMake(‘Audi’) x;
```

Stored Procedures and User-defined Functions: PL/pgSQL

```
CREATE OR REPLACE PROCEDURE transaction_test(k int)
LANGUAGE plpgsql
AS $$
DECLARE l integer = 1;
BEGIN
    FOR i IN k..k+10 LOOP
        INSERT INTO test1 (a) VALUES (i);
        IF i % 2 = 0 THEN
            COMMIT;
            RAISE NOTICE ‘executed commit no %’, l;
            l := l+1;
        ELSE
            ROLLBACK;
        END IF;
    END LOOP;
END;
$$;
```