ÜBUNGSBLATT 2

Richard Fussenegger

Aufgabe 1 – Gespeicherte Programme

1A. Gespeicherte Funktion

```
DELIMITER //

CREATE FUNCTION formattime(i INTEGER, s TINYTEXT) RETURNS TINYTEXT DETERMINISTIC
BEGIN
  DECLARE _r TINYTEXT;

  IF 1 > i     THEN SET _r := "";
  ELSEIF 1 = i THEN SET _r := CONCAT(i, " ", s);
  ELSE              SET _r := CONCAT(i, " ", s, "s");
  END IF;

  RETURN _r;
END//

CREATE FUNCTION flighttime(arrival DATETIME, departure DATETIME) RETURNS TINYTEXT DETERMINISTIC
BEGIN
  DECLARE _t TIME;
  DECLARE _h, _m TINYTEXT;

  SET _t := TIMEDIFF(arrival, departure);
  SET _h := formattime(HOUR(_t), "hour");
  SET _m := formattime(MINUTE(_t), "minute");

  RETURN CONCAT(_h, IF("" = _h, "", " "), _m);
END//

DELIMITER ;
```

1B. Gespeicherte Prozedur

```
DELIMITER //

CREATE PROCEDURE book_asap(
  IN iata_from CHAR(3),
  IN iata_to CHAR(3),
  IN earliest DATETIME,
  IN pnr CHAR(9)
) DETERMINISTIC
BEGIN
  DECLARE _fid INT(11);
  DECLARE _fnr CHAR(8);
  DECLARE _pid INT(11);

  SELECT `passagier_id`
  INTO _pid
  FROM `FlughafenDB`.`passagier`
  WHERE `passnummer` = pnr
  LIMIT 1;
  IF _pid IS NULL THEN
    SIGNAL SQLSTATE "45000" SET MESSAGE_TEXT = "Sorry, but I don't know you";
  END IF;

  SELECT `flug`.`flug_id`, `flug`.`flugnr`
  INTO _fid, _fnr
  FROM `FlughafenDB`.`flug`
    INNER JOIN `FlughafenDB`.`flughafen` AS `f1`
      ON `f1`.`flughafen_id` = `flug`.`von`
    INNER JOIN `FlughafenDB`.`flughafen` AS `f2`
      ON `f2`.`flughafen_id` = `flug`.`nach`
  WHERE `f1`.`iata` = iata_from
    AND `f2`.`iata` = iata_to
    AND (
      SELECT
        `flugzeug`.`kapazitaet` - COUNT(*)
      FROM `FlughafenDB`.`buchung`
        INNER JOIN `FlughafenDB`.`flug` AS `sf`
          ON `sf`.`flug_id` = `buchung`.`flug_id`
        INNER JOIN `FlughafenDB`.`flugzeug`
          ON `flugzeug`.`flugzeug_id` = `sf`.`flugzeug_id`
      WHERE `buchung`.`flug_id` = `flug`.`flugzeug_id`
      GROUP BY `flugzeug`.`kapazitaet`
    ) > 0
    AND `flug`.`abflug` > CAST(earliest AS DATETIME)
  ORDER BY `flug`.`ankunft` ASC
  LIMIT 1;

  IF _fid IS NULL THEN
    SIGNAL SQLSTATE "45000" SET MESSAGE_TEXT = "Sorry, no flight available", MYSQL_ERRNO = 1644;
  END IF;

  INSERT INTO `buchung` SET
    `flug_id`      = _fid,
    `passagier_id` = _pid,
    `preis`        = 1e5
  ;

  SELECT CONCAT(
    "A seat from ", iata_from, " to ", iata_to, " was booked on flight ", _fnr
  ) AS `booked`;
END//

DELIMITER ;
```

1C. Gespeicherte rekursive Prozedur

```
DELIMITER //

CREATE PROCEDURE erreichbare_flughaefen(
  IN airport_id INT(11),
  IN max_hops INT,
  IN rlevel INT
) DETERMINISTIC
BEGIN
  DECLARE _done BOOLEAN DEFAULT FALSE;
  DECLARE _aid INT;
  DECLARE _cur CURSOR FOR SELECT `flughafen_id` FROM `reachable_airports` WHERE `hops` = rlevel;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET _done := TRUE;

  IF 0 = rlevel THEN
    SET max_sp_recursion_depth = 255;
    DROP TABLE IF EXISTS `reachable_airports`;
    CREATE TABLE `reachable_airports` (
      `flughafen_id` INT,
      `hops` INT,
      PRIMARY KEY(`flughafen_id`, `hops`)
    )
    ENGINE = MEMORY;
  END IF;

  INSERT INTO `reachable_airports` (`flughafen_id`, `hops`)
  SELECT `nach`, rlevel FROM `FlughafenDB`.`flugplan` WHERE `von` = airport_id AND (
    SELECT `hops`
    FROM `reachable_airports` AS `t`
    WHERE `t`.`flughafen_id` = `flugplan`.`nach`
      AND `t`.`hops` = rlevel
  ) IS NULL;

  IF rlevel <= max_hops THEN
    OPEN _cur;
    SET rlevel := rlevel + 1;
    _l: LOOP
      FETCH _cur INTO _aid;
      IF _done THEN
        LEAVE _l;
      END IF;
      CALL erreichbare_flughaefen(_aid, max_hops, rlevel);
    END LOOP;
    CLOSE _cur;
  END IF;
END//

DELIMITER ;
```

Aufgabe 2 – Triggers und Views

2A. View

```
CREATE TABLE IF NOT EXISTS `buchung` LIKE `FlughafenDB`.`buchung`;

CREATE OR REPLACE VIEW `buchungsview`
AS SELECT * FROM `buchung` WHERE `flug_id` BETWEEN 10 AND 1e3
WITH CHECK OPTION;
```

Die WITH CHECK OPTION-Klausel kann einer aktualisierbaren Ansicht gegeben werden um zu verhindern, dass Einträge hinzugefügt werden die nicht der WHERE-Klausel der SELECT-Anweisung der Ansicht entsprechen. Beim Versuch einen Eintrag einzufügen der ungültig ist wird der SQLSTATE 1369 zurückgegeben mit der Meldung „CHECK OPTION failed 'viewname'".

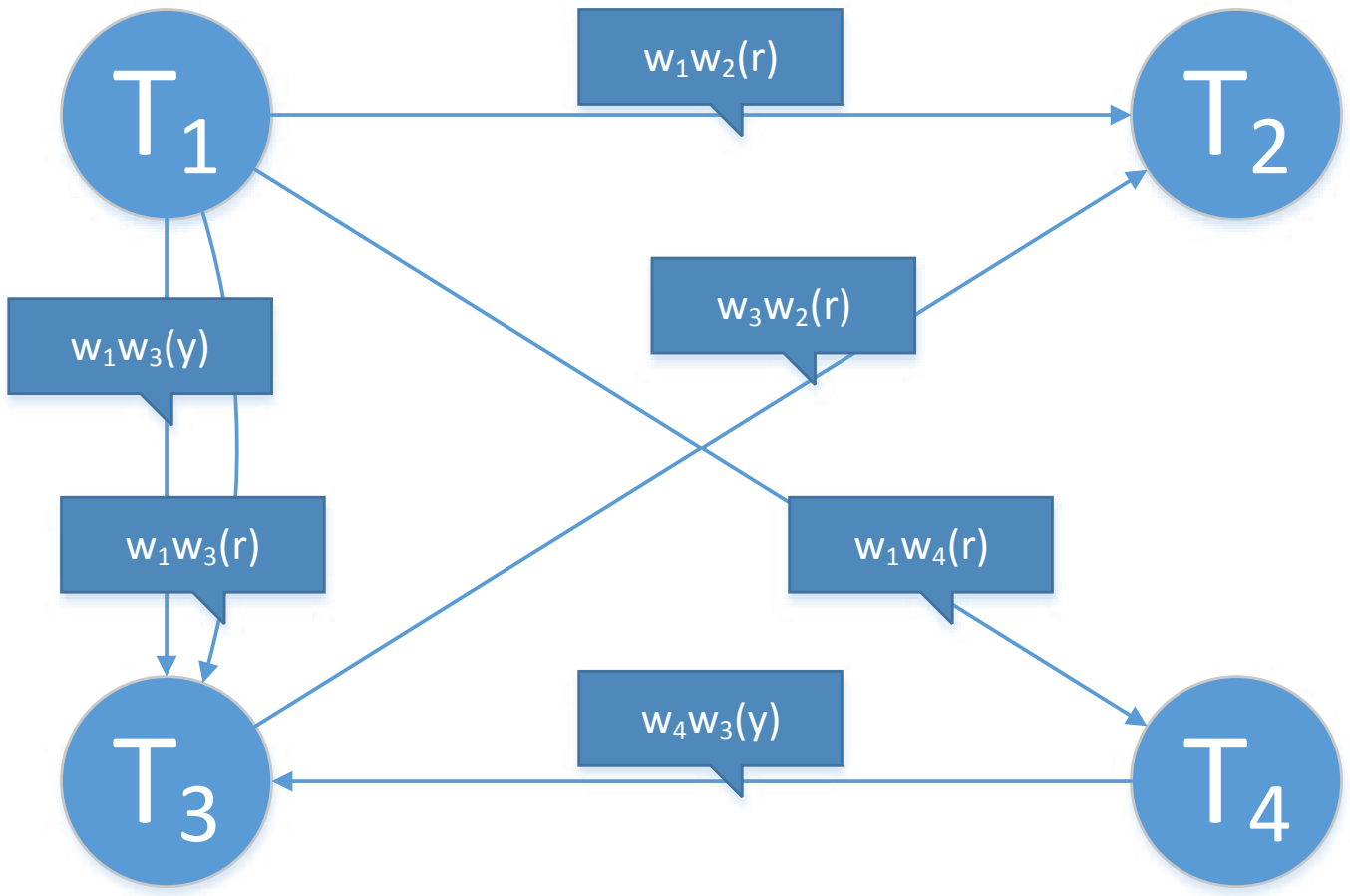
2B. Trigger

```
DELIMITER //

CREATE TRIGGER `money` BEFORE INSERT ON `buchung` FOR EACH ROW
BEGIN
  IF NEW.`preis` <= 0.0 THEN
    SET NEW.`preis` = 1e5;
  END IF;
END//

DELIMITER ;
```

Aufgabe 3 – Transaktionen

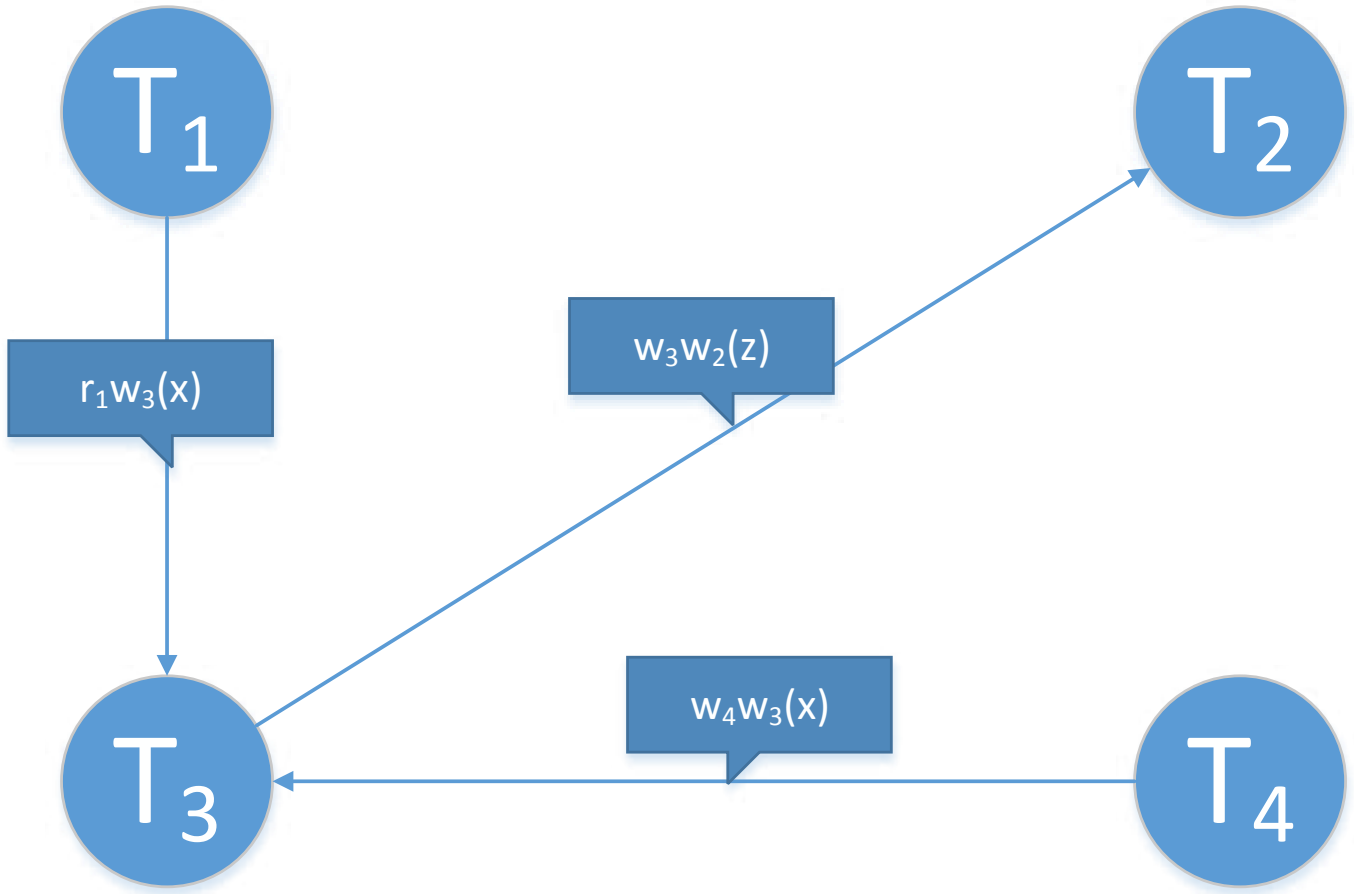
3A. Mögliche Ausführung: $T_2\ T_3\ T_4\ T_1$

$r_3(x)$
$r_2(y)$
$w_3(x)$
$r_4(x)$
$w_3(y)$
$w_4(y)$ ... muss auf $w_3(y)$ warten
$r_4(z)$
$w_1(z)$
$w_1(y)$ ... muss auf $w_4(y)$ und $w_3(y)$ warten
$w_2(r)$
$w_3(r)$ ... muss auf $w_2(r)$ warten
$w_1(r)$ ... muss auf $w_3(r)$ und $w_2(r)$ warten



3B. Mögliche Ausführung: $T_2\ T_3\ T_4\ T_1$

$r_1(z)$
$r_3(z)$
$r_4(z)$
$r_2(y)$
$w_2(z)$
$w_3(x)$
$r_1(x)$ ... muss auf $w_3(x)$ warten
$w_4(x)$ ... muss auf $w_3(x)$ warten
$w_4(y)$
$w_3(z)$ ... muss auf $w_2(z)$ warten



Aufgabe 4 – 2-Phasen-Sperrprotokoll

T_a	T_b	T_c	
		`r(x)`	`x++ (1)`
`w(y)`			`SET WRITE LOCK y`
`r(x)`			`x++ (2)`
	`w(x)`		`WAIT` (Unterbrechung von T_b)
		`r(x)`	`x++ (3)`
		`r(z)`	`z++ (1)`
		`w(z)`	`SET WRITE LOCK z`
	`w(y)`		
		`w(x)`	`WAIT`
`r(y)`			`y++ (1)`
`eot`			löst alles auf von T_a (Reevaluierung T_b)
		`w(x)`	`SET WRITE LOCK x`
	`w(x)`		
		`eot`	

T_b wartet auch nach dem `eot` von T_a weiter, da auch T_c eine Lesesperre auf `x` erstellt hat, entsprechend wird T_b immer wieder unterbrochen. Das führt dazu, dass T_a problemlos fertig wird und danach T_c problemlos fertig wird. T_b kann erst nach T_a und T_c ausgeführt werden.