Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Datenbanksysteme · Prof. Dr. Grust

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Datenbanksysteme I

WS 2021/22
Torsten Grust, Christian Duta, Tim Fischer

Assignment #7

Submission Deadline: December 15, 2021 - 10:00

Please note that students will have the opportunity to **evaluate lectures**. Please help us to improve **your** courses by providing precious feedback. Check your Mailbox ~~on December 6th 2021~~ **in the coming days**.

Exercise 1: Constraints and References (6 Points)

Answer the following questions **briefly**:

1. Explain the terms **foreign key constraint**, **inclusion constraint**, and **referential integrity**.
2. What are differences between value-based references and pointers?
3. List all kinds of database constraints that were discussed in the lecture so far.

Exercise 2: Cascading Deletes (6 Points)

Create two SQL tables `r` and `s` along with their instances and constraints.

At creation, instances of `r` and `s` exhibit the following properties:

- No value in a key column must be referenced by more than one value of a foreign key column.
- Instances of both `r` and `s` must contain at least 5 rows.

Once the tables have been created, deletions in `r` or `s` show the following behavior:

- If one arbitrary row from `s` is deleted, it must result in both `r` and `s` being empty.
- If one arbitrary row from `r` is deleted, it must result in `r` being empty (while `s` remains unchanged).

Hand in a list of SQL commands that create tables, instances, and constraints with these properties and two SQL `DELETE` statements that show the previously described behavior.

Exercise 3: Existential Quantification (9 Points)

Assume tables

```
CREATE TABLE p (x int);
CREATE TABLE q (x int);

ALTER TABLE p ALTER COLUMN x SET NOT NULL;
ALTER TABLE q ALTER COLUMN x SET NOT NULL;
```

Implement SQL queries to simulate the following **set operations**:

1. Set difference `p` $\setminus$ `q`
2. Set intersection `p` $\cap$ `q`
3. The subset relation $\subseteq$: Implement a query that outputs (a single-row, single-column table containing) `true` iff `p` $\subseteq$ `q`.

As these are set operators, the result should not contain duplicates. Formulate your queries using predicates `(NOT) IN` and `(NOT) EXISTS` and include test cases in your answer.

Important: Do **not** use the corresponding SQL set operators `INTERSECT`, `EXCEPT` and `UNION`.

Exercise 4: Intra-Table Foreign Keys (9 Points)

We have discussed a particular relational encoding of tree-shaped data structures (slide 12, chapter 7 "Referential Integrity"). In this encoding, each node uses a foreign key value to refer to its parent.

Alternatively, we may want each node to reference its children instead. We assume that nodes in the tree have at most two children. Create a table for this alternative encoding, with appropriate constraints. Encode the tree structure from slide 12 appropriately and store it in your table.

Next, formulate SQL queries on your tree representation to answer the following questions:

1. What is the label of the sibling of the node with label `'E'`?
   Output: a table with single column `sibling` of type `label`. The output is empty, if the node with label `'E'` has no sibling or does not exist.
2. What are the labels of the grandchildren of the node with label `'A'`?
   Output: a table with single column `grandchildren` of type `label`. The output is empty, if the node with label `'A'` does not exist.
3. What is the label of the root node?
   Output: a table with single column `root` of type `label`. The output is empty, if the tree is empty.