EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Database Systems 1
Summerterm 2023
Prof. Torsten Grust, Louisa Lambrecht, Tim Fischer
WSI — Database Systems Research Group

Assignment 4

Hand in this assignment until **Thursday, 25 May 2023, 12:00** at the latest.

Ⓔxam-style Exercises

Exercises marked with Ⓔ are similar in style to those you will find in the exam. You can use these to hone your expectations and gauge your skills.

Running out of ideas?

Are you hitting a roadblock? Are some of the exercises unclear? Do you just need that one hint to get the ball rolling? Refer to the `#forum` channel on our Discord server and check the tag for this assignment—maybe you'll find just the help you need.

Task 1: Types (1 credit)

A deck of French-suited playing cards—commonly referred to as a bridge or poker deck—comprises 52 cards. Each of the four suits C(lubs), D(iamonds), H(earts), and S(pades) features 13 ranks: 2, 3, 4, 5, 6, 7, 8, 9, 10, J(ack), Q(ueen), K(ing) and A(ce). Both suits and ranks are given in ascending significance in a game of poker.

In contrast, a piquet deck—as used for games of skat—only consists of 32 cards comprising the French-suited cards ranked seven through ace. Some games require adding a joker to the deck, increasing the card count to 33. A joker has a rank (denoted as Jo) but no suit.

Note: Hand in everything!

Please add all of your *SQL* statements—whether or not they result in errors—into a single *SQL* file and hand it in.

(a) First we will create two poker decks, `CARDS_BAD` and `CARDS_GOOD`. Please follow the steps below and execute your queries on a Postgre*SQL* database system.

i. Construct a `CREATE TABLE` statement for a table `CARDS_BAD` with two columns: `suit` (`CHAR(1)`) and `rank` (`VARCHAR(2)`).

ii. Write `INSERT` statements to fill `CARDS_BAD` with a complete poker deck (52 rows). Adding multiple rows with a single `INSERT` statement may come in handy. This *bulk insert* method has been introduced in slide 35 in slide set `db1-04.pdf`. Check if the data was inserted correctly using the `TABLE` command.

iii. Now, create a second table `CARDS_GOOD`. This time use `CREATE TYPE` to create dedicated data types `suits` and `ranks` for columns `suit` and `rank` that enforce restricted domains *dom*(`suits`) and *dom*(`ranks`). The domains **must** allow French-suited playing cards **only**!

iv. Again, insert a complete poker deck into `CARDS_GOOD`. Think about reusing your created `INSERT` statements.

(b) Use the `DELETE FROM ⟨table⟩ AS t WHERE t.rank < '7'` statement to convert the created poker decks in `CARDS_BAD` and `CARDS_GOOD` to skat compatible decks (without jokers). Are the resulting tables in both cases as expected? Explain your results.

(c) Try to `INSERT` a joker into both decks. Do not modify your type definitions! You will encounter difficulties. Please explain your results.

Task 2: Relation Schema vs. Instance (1 credit)

Imagine you want to plan the chore chart of your living community using an *RDBMS*. The chart is expected to provide an assignment of the services TRASH, KITCHEN and BATHROOM to the flatmates *Annika*, *Pierre* and *Leonie* on a weekly basis. The relational model implies that your chart is represented in a tabular form. Figure 1 shows three possible variants of a CHART relation.

CHART1

week	*Annika*	*Pierre*	*Leonie*
49	TRASH & KITCHEN	null	BATHROOM
50	BATHROOM	TRASH	KITCHEN

CHART2

week	TRASH	KITCHEN	BATHROOM
49	*Annika*	*Annika*	*Leonie*
50	*Pierre*	*Leonie*	*Annika*

CHART3

week	name	service
49	*Annika*	TRASH
49	*Annika*	KITCHEN
49	*Leonie*	BATHROOM
50	*Annika*	BATHROOM
50	*Pierre*	TRASH
50	*Leonie*	KITCHEN

Figure 1: Exemplary possibilities to represent a chore chart in an RDBMS

In this task we will study the properties of these three variants in terms of their *relation schemas* and *relation instances*.

(a) For each of the CHART relations, write down its *relation schema* and *relation instance*. Use the notation introduced in the lecture (slide 23 and slide 28 in slide set `db1-04.pdf`).

(b) Construct `CREATE TABLE` statements for each of the displayed representations. For all columns, choose a data type which is as precise as possible, but puts no constraints on names of new flatmates or services.

(c) Ⓔ Explain what changes to the *schema* and/or *instance* are needed for every relation, if we want to:

i. add the plan for week 51 (*Annika*: KITCHEN, *Pierre*: BATHROOM, *Leonie*: TRASH)

ii. add an additional service COOK for *Pierre* in week 50

iii. switch *Leonie* with a new flatmate *Adrian*.

In the relational model, relation schemas are assumed to be stable while instances change frequently. Given this, which relation is the best choice to represent the chore chart?

(d) Specify *SQL* `INSERT`, `UPDATE` and `DELETE` statements for those relations in subtasks i. to iii. that only need their instance changed.