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

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Datenbanksysteme I

WS 2019/20
Torsten Grust, Christian Duta

Assignment #4

Submission Deadline: November 19, 2019 - 10:00

Exercise 1: Types **(10 Points)**

A poker card deck consists of 52 cards. Each of the four suits ♣, ♠, ♡, ♢ features 13 ranks: 2, 3, 4, 5, 6, 7, 8, 9, *J*(*ack*), *Q*(*ueen*), *K*(*ing*), 10, *A*(*ce*). Both, suits and ranks are given in ascending significance.

In contrast a skat card deck comprises 32 cards only. It contains the same cards as the poker card deck, but ranks lower than seven are missing. Some games require that a joker is added to the skat card deck, increasing the deck to 33 cards. A joker does not have a suit, but only its rank denoted as *Jo*.

Note: Please insert all of your SQL statements – whether or not they result in errors – into a single SQL file and hand it in.

1. First we will create two poker card decks, `CARDS_BAD` and `CARDS_GOOD`. Please follow the steps below and execute your queries on a PostgreSQL database system.
   (a) Construct a `CREATE TABLE` statement for a table `CARDS_BAD` with two columns: `suit` (`CHAR(1)`) and `rank` (`VARCHAR(2)`).
   (b) Write `INSERT` statements to fill `CARDS_BAD` with a complete poker card deck (52 rows). Adding multiple rows with a single `INSERT` statement may come in handy. Using this *bulk insert* method has been introduced on slide 38. Check if the data was inserted correctly using the `TABLE` command.
   (c) Now, create a second version of the table `CARDS_BAD`. This time name it `CARDS_GOOD` and 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 poker card sets **only**!
   (d) Again, insert a complete poker card deck into `CARDS_GOOD`. Think about reusing your created `INSERT` statements.
2. Use the `DELETE FROM <table> t WHERE t.rank < '7'` statement to convert the created poker card decks in `CARDS_BAD` and `CARDS_GOOD` to skat card decks (without jokers). Insert the queries in your SQL file. Are the resulting tables in both cases as expected? Explain your results.
3. Try to `INSERT` a joker into both card decks. Do not modify your type definitions! You will encounter difficulties. Please explain your results. Note the queries in your SQL file.

Exercise 2: Relation Schema vs. Instance **(20 Points)**

Imagine you want to plan the chore chart of your living community using a *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. However, Figure 1 shows three possible variants of a `CHART` relation.

CHART_1

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

CHART_2

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

CHART_3

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 a RDBMS

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

1. For each of the `CHART` relations, write down its *relation schema* and *relation instance*. Use the notation introduced in the lecture (slides 23 and 28 in chapter "The Relational Data Model").
2. 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.
3. Explain what changes to the *schema* and/or *instance* are needed for every relation, if we want to:
   (a) add the plan for week 51 (*Annika*: **KITCHEN**, *Pierre*: **BATHROOM**, *Leonie*: **TRASH**)
   (b) add an additional service **COOK** for *Pierre* in week 50
   (c) 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?
4. Specify SQL `INSERT`, `UPDATE` and `DELETE` statements for 3a, 3b and 3c for those relations which only need their instance changed.