UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Exam

DIT032 / DAT335 – Data Management

Tuesday, June 11th, 2019 08:30 - 12:30

Examiner:
Philipp Leitner

Contact Persons During Exam:
Philipp Leitner (+46 733 05 69 14)
Joel Scheuner (+46 733 42 41 26)
(visitations around 09:30 and 10:30)

Allowed Aides:
None except English dictionary (non-electronic), pen/pencil, ruler, and eraser.
Check the back of your exam papers for additional help in the appendix.

Results:
Exam results will be made available no later than in 15 working days through Ladok.

Grade Limits:
For GU students: 0 – 49 pts: U, 50 – 84 pts: G, 85+ pts: VG
For Chalmers students: 0 – 49 pts: U, 50 – 69 pts: 3, 70 – 84 pts: 4, 85+ pts: 5

Review:
The exam review will take place Friday, July 5th (10:00 - 11:00) in room 473 (Jupiter building, 4th floor). Please refer to Canvas for potential updates.

Task 1 – Theory and Understanding (25 pts)

(every question can and should be answered in a few sentences, plus an example if asked for)

Q1.1: We discussed three different ways to map inheritance structures from EER to the relational model. Name and describe them. What are the advantages and disadvantages of each strategy? *9 pts*

Q1.2: Briefly describe the Master/Slave replication strategy. *3 pts*

Q1.3: Describe the three-value logic used by SQL. How does it differ from the more traditional boolean logic, and in what context is it used? *2 pts*

Q1.4: What is a Key/Value datastore? Briefly describe the key principles. *2 pts*

Q1.5: What is "well-formedness" in the context of XML or JSON? How does it relate to validity? Provide 2 important syntactical rules for both, JSON and XML, which need to be fulfilled to make a document well-formed. *6 pts*

Q1.6: What problem do views in SQL solve? Which different types of views are there? *3 pts*

Task 2 – EER Diagrams (22 pts)

Consider the following excerpt of the domain description for the database of a blogging platform. Model the described domain using an EER diagram with the notation we used in the course (find a cheat sheet in the appendix of your exam papers). Use the 1,N,M notation for describing cardinalities rather than the min-max notation.

Blogging Platform:
The database needs to keep track of blogs. Each blog is identified through a unique name as well as a unique URL. Blogs also have a description. Each blog contains zero to many posts. Each post has a running number (which is only unique for a specific blog), a title, an article text, a publishing date, and a flag that signifies whether the post has already been made public. Multiple posts are allowed to have the same title. Posts may also embed arbitrarily many media items (images, videos, etc.). Each media item may be embedded in zero to many posts, and is identified through a unique ID. Further, we need to store the type as well as the content of each media item.

In addition to blog posts, the database also needs to store different types of users. Each user has a unique username, a real name (consisting of first and last name), and a password hash. We need to distinguish between free and paying users. For the latter, we additionally need to store credit card information, which consists of a card number and a validity date.

Each user may further be a blog author and/or a commenter. Authors may own many blogs, but they need to own at least one. They may also upload zero to many media items, and we need to store the upload date for each media item. Finally, authors also write posts. Each post is written by at least one author, but sometimes multiple authors may collaborate on the same post. For these cases we need to store who the lead author is. Every author also has to be a paying user.

Users who are commenters have written at least one comment. A comment is associated to exactly one post, but of course a single commenter can write many comments to many different posts. Similar to posts, comments are identified through a running number that is only unique in the context of the post that they relate to. Further, a comment consists of a title and text (both not necessarily unique). For each post we need to store how many comments there are. Note that commenters may be free or paying users, and commenters may or may not also be authors.

Task 3 – Partial Relational Mapping and Normalization (8 pts)

Provide a relational model using the notation we used in the course (see also Task 4 for an example) for the two entities `Blog` and `Post` from Task 2. Ensure that all relations have a primary key, define foreign keys as necessary, and make sure that all attributes from the relevant parts of the textual description in Task 2 are mapped. You *may not* introduce any artificial primary keys in this task.

Provide two versions of the relational model – one that is in 3NF, and one that has at least one unnecessary functional dependency (i.e., one that is *not* in 3NF).

Task 4 – Relational Algebra (20 pts)

Relational Model:
HUMAN(first_name, last_name, birthday)

PET(name, type, age)

HOUSE(address, year_built)

PET_OWNER(first_name, last_name, pet)
{first_name,last_name} → {HUMAN.first_name,HUMAN.last_name}
pet → PET.name

PET_LIVES_WITH_AT(address, first_name, last_name, pet, since)
address → HOUSE.address
{first_name,last_name} → {HUMAN.first_name,HUMAN.last_name}
pet → PET.name

Notes: First_name and last_name are a composed foreign key pointing at the composed primary key of HUMAN. Address and pet are also foreign keys pointing at the primary keys of HOUSE and PET, respectively. The attribute "since" indicates since when a pet lives at a given address with a given human.

Given this relational model, write relational algebra statements that exactly represent the following queries. Use the mathematical notation from the course (for the correct notation you can again refer to the appendix).

Queries:

Q4.1: Return since when the pet with the name "Fifi" is living with "Philipp Leitner" at the house with the address "21 Jump Street".

Q4.2: For each pet with the type "dog", return the name of the pet as well as the first name, last name, and birthday of the owner.

Q4.3: Return a list of addresses and how many pets live at this address. *Hint:* Consider that a single pet may live with multiple humans at any address.

Q4.4: Return a list of all humans (first and last names) as well as the names of the pets they own. Humans that do not own any pets should still be contained in the list.

Task 5 – SQL (25 pts)

Given the same relational model as for Task 4, write the following SQL statements or queries.

Statements:

Q5.1: Return a list of all humans (all attributes), ordered in descending order by their age.

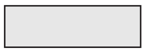
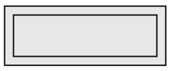
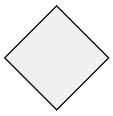
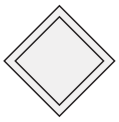
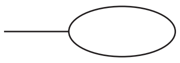
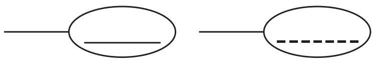
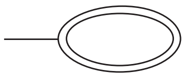
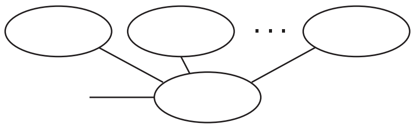
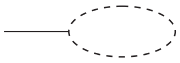
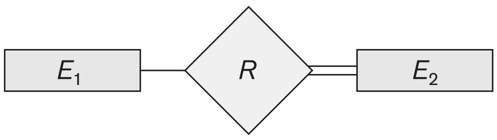
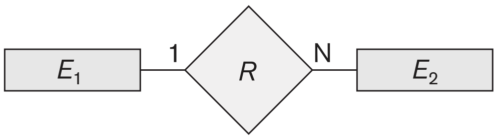
Q5.2: Return a list of all houses, the year that they have been built in, and the pets that live in them. *Hint:* ensure that only a single row per house and pet is returned, even if multiple owners live in the same house with the same pet.

Q5.3: Calculate the average age of all pets with the type “cat”.

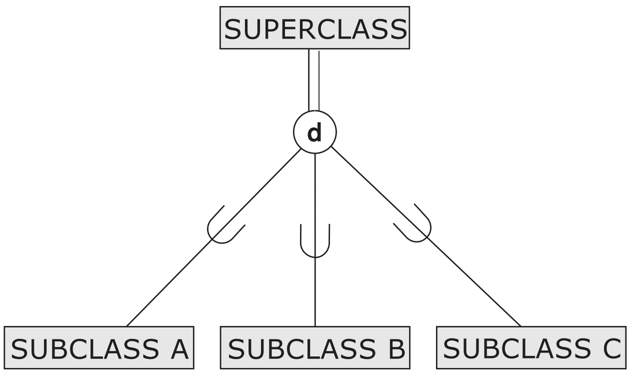
Q5.4: Find the names of all pets that do not (only) live with their owners.

Q5.5: Find the names of all pets that have the same name as the first name of a human.

Appendix: Notation Guidelines for EER and RA

Symbol	**Meaning**
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute / Dashed Underline for Partial Key
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of E_2 in R
	Cardinality Ratio 1: N for E_1 : E_2 in R

Total Disjoint Specialization



Partial Overlapping Specialization

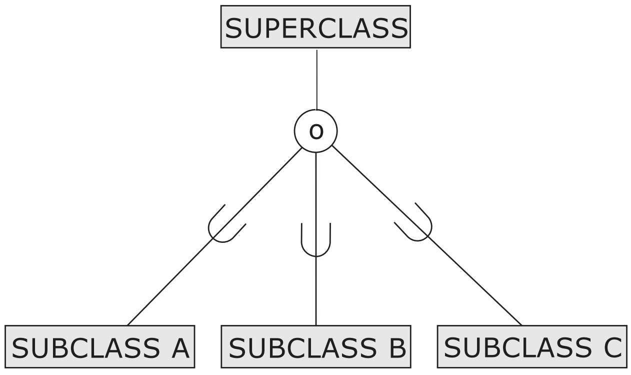


Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{\text{(<join attributes 1>),}}$ $_{\text{(<join attributes 2>)}} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{\text{(<join attributes 1>),}}$ $_{\text{(<join attributes 2>)}}$ R_2 OR $R_1 * R_2$
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R$_1$ in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

`<grouping>`$\mathscr{F}$`<functions>(R)`

whereas `<functions>` is a list of

```
[MIN|MAX|AVERAGE|SUM|COUNT] <attribute>
```