UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Exam

DIT032 / DAT335 – Data Management

Monday, March 18th, 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, April 5th (09:00 - 10:00) in room 473 (Jupiter building, 4th floor). Please refer to Canvas for potential updates.

Task 1 – Theory and Understanding (22 pts)

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

Q1.1: Define the idea of meta-data. Give examples using technologies we introduced in the lecture. *4 pts*

Q1.2: List and describe the transaction primitives in SQL transactions. *4 pts*

Q1.3: What are “prepared statements” in JDBC? What are they useful for? Describe it briefly in your own words. *4 pts*

Q1.4: Explain the concept of sharding in a distributed database. What are the properties of a good sharding function? *4 pts*

Q1.5: Shortly describe the structure of a well-formed JSON document. Provide a minimal example that uses all core features of JSON (at least one of each: simple field, complex field, list). *6 pts*

Task 2 – EER Diagrams (21 pts)

Consider the following excerpt of the domain description for the database of an online store. 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. If something is not specified, make a reasonable assumption and note it down in plain text.

Online Store:
The database needs to keep track of products. Products are identified through a product ID, and also have an unique name. Products also need to store a product description, which is not necessarily unique. Finally, products are classified into product groups, so that each product is in exactly one product group and each product group contains one or more products. A product group has an unique name, and we need to have access to how many products are in each product group.

The database also needs to store two types of persons: customers and employees. Each person can be either a customer, an employee, or both. For each person we need to store an unique ID, an address (consisting of street name, house number, post code, and country code), and a name consisting of first, middle, and last name. For customers, we additionally store a customer ID. A subset of customers are premium customers, for which we need to store a fixed discount amount that is subtracted from every order (e.g., 5%). Employees have an employee ID, a job role, and a salary.

Customers place orders in the system. Orders are identified through an unique order number, and can contain multiple products (but at least one). Of course, the same product can be ordered many times (and some products have never been ordered so far). Some products may be contained multiple times in the same order, so we need to track how often any given product is contained in a given order.

Finally, employees manage product groups. Each employee is responsible for zero to many product groups, and each product group is managed by exactly one employee.

Task 3 – Mapping EER to the Relational Model (12 pts)

Consider the EER model below. Construct a relational model that represents this domain. Select primary keys and introduce foreign keys as necessary. Use the notation that we used in the course to indicate relations, attributes, primary keys, and foreign keys (see Task 4 for an example of the required notation).

```
../img/books_shelves.pdf
```

Task 4 – Relational Algebra (20 pts)

Relational Model:
COMPANY(<u>shortname</u>, full_name)

BUILDING(<u>name</u>, <u>city</u>, owner)
owner → COMPANY.shortname

ROOM(<u>roomname</u>, capacity, building_name, building_city)
{building_name,building_city} → {BUILDING.name, BUILDING.city}

(owner is a foreign key pointing at COMPANY.shortname, building_name and building_city together point at the composed primary key of BUILDING)

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: Find the room name and capacity of all rooms with a capacity of more than 50 persons.

Q4.2: List all attributes of all rooms in buildings owned by the company with the full name "ACME Inc".

Q4.3: How many buildings does each company own? Provide a list of company full names and how many buildings are owned by this company.

Q4.4: Provide a list of all pairs of buildings which are in the same city, but owned by different companies. It is ok if the same pairs appear multiple times 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: Create the table `BUILDING`. Assume that a table `COMPANY` with the schema defined in Task 4 already exists. Select suitable data types. If a company is deleted, all of its buildings should also automatically be deleted from the database.

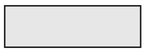
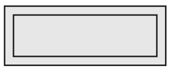
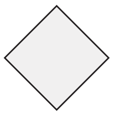
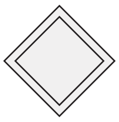
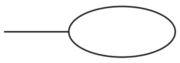
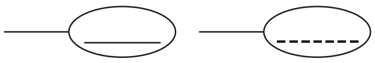
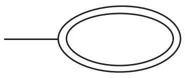
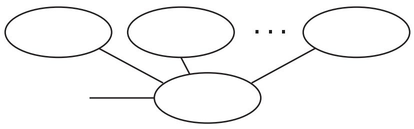
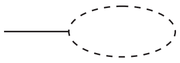
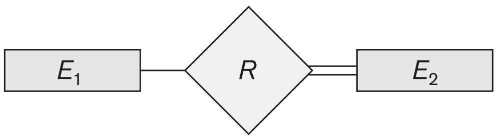
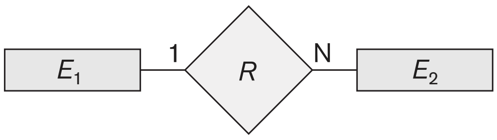
Q5.2: Return all buildings in one of the following cities: Gothenburg, Stockholm, Copenhagen, Oslo, or Helsinki. Do *not* use more than a single conditional statement in the "where" clause of the query. Sort by the short name of the owner, in ascending order.

Q5.3: Return a list of all company full names, the buildings that they own, and the rooms in these buildings (one return row per room). Companies not owning any buildings should still be contained in the result (with NULL values for all attributes of buildings and rooms).

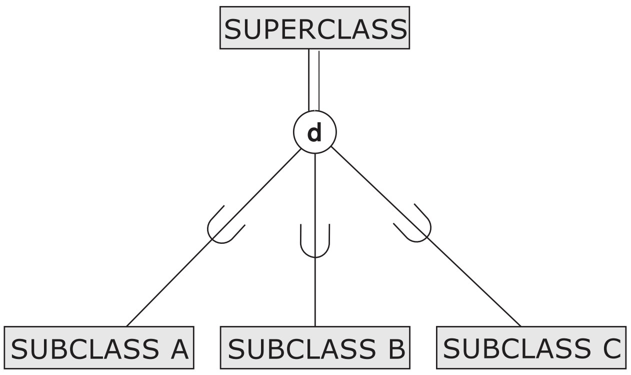
Q5.4: Calculate the average capacity of rooms in the city "Gothenburg".

Q5.5: Return the names of all rooms which have a higher capacity than the room with the name "Alfons".

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

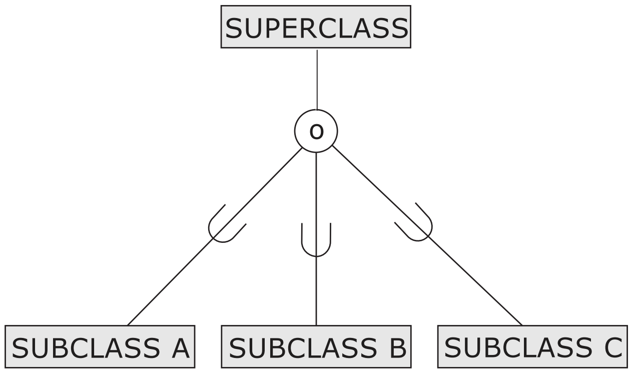


Table 8.1 Operations of Relational Algebra

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 $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

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

whereas `<functions>` is a list of

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