UNIVERSITY OF GOTHENBURG

CHALMERS UNIVERSITY OF TECHNOLOGY

Exam

DIT033 / DAT335 – Data Management

Monday, March 14, 2022, 08:30 - 12:30

Examiner:
Philipp Leitner

Contact Persons During Exam:
Nayla Nasir (+46 31 772 1986)
(visitations around 9:30 and 11:00)

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 – 49.9 pts: U, 50 – 69.9 pts: G, 70–84.9 pts: VG, 85+ pts: MVG
For Chalmers students: 0 – 49.9 pts: U, 50 – 69.9 pts: 3, 70 – 84.9 pts: 4, 85+ pts: 5

Review:
The exam review will take place Friday, April 8th (11:00 - 12:00). Please refer to Canvas for potential updates.

Task 1 – Theory and Understanding (24 pts)

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

Q1.1: Explain Cartesian product with an example (you can use two tables, each having atleast 3 rows). What is the basic difference between a Cartesian product and a join? Explain how in relational algebra, a join operation can be specified as a Cartesian product using a sequence of operations.(5 pts)

Q1.2: What is normalization? Explain 3NF with an example? (2 pts)

Q1.3: Why indices are used in SQL? Write one advantage and one disadvantage of using an index. (3 pts)

Q1.4: What is ”well-formednes” in the context of XML or JSON? How does it relate to validity? Provide two important syntactical rules for XML, which need to be fulfilled to make a document well-formed.(4 pts)

Q1.5: Explain the concept of ”non repeatable reads’. What are different transaction isolation levels, and which of the transaction isolation violations, each of these addresses? (you can illustrate in tabular form). (5 points).

Q1.6: Explain how replication is useful in terms distributed databases? What are two types of replication? How replication differs from sharding? (5 pts)

Task 2 – EER Diagrams (20 pts)

Consider the following excerpt of the domain description for a library database. 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.

Libraries:

The database tracks (public) libraries in different cities. A library has an unique library identifier and a name, which is not necessarily unique. Further, the system needs to keep track of how many books are stored in each library.

Every library holds many books, and each book is available in 0 to many libraries. A book is identified by an unique ISBN number. Books further have a title, page count, and need to store whether the book is hard- or soft-cover. Books are available in different editions. Every edition is identified through their publishing year, which is unique for a book (but not unique overall). Every book has at least one edition.

The system also needs to store information about persons interacting with the libraries. A person has an unique social security number ("ssn"), and a first name, last name, as well as a date of birth. Every person lives in exactly one city. Libraries are also located in cities, but whereas cities naturally may be home to many persons, they may only contain a single (optional) library. Cities are identified by a combination of name and region.

Special types of persons are authors and customers. Every author has written at least one book, but they may have written many. Conversely, a book can have many authors, but needs to have at least one. Book authors do not change between editions. Customers have a customer number and rent editions of books. Every customer may rent any number of book editions, and each edition may be rented by many customers. Note that any person in the database can be an author, a customer, neither, or both.

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

Consider the EER model for Task 2. 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).

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

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 “lulu” is living with “John Smith” at the house with the address “vanilijgatan 11”.

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 (20 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 have the same name as the first name of a human.

Task 6 – MongoDB (4 pts)

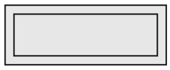
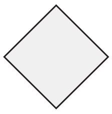
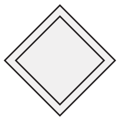
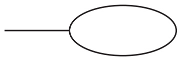
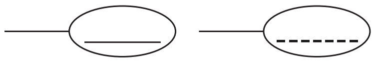
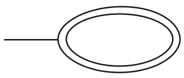
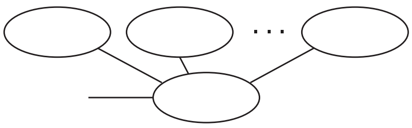
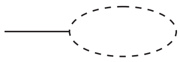
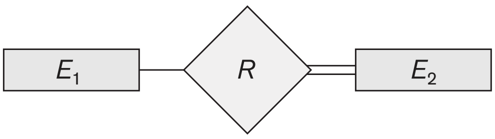
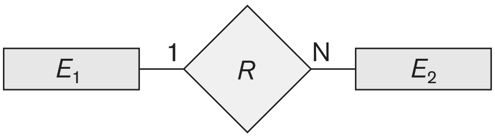
Explain in plain English what the following statements do when executed in the MongoDB shell. Be as exact as possible.
Note that for this task it is not necessary to know what the content of the database is.

Statements:

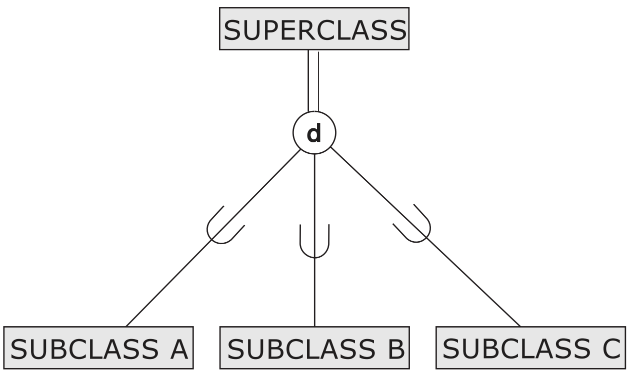
Q6.1:
```
db.students.find(
    { "student.name" : "James Lewis" },
    { "student.id" : true }).pretty()
```

Q6.2:
```
db.students.update(
    { "student.name" : "James Lewis" },
    { "$set" : { "student.enrolled" : false } })
```

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

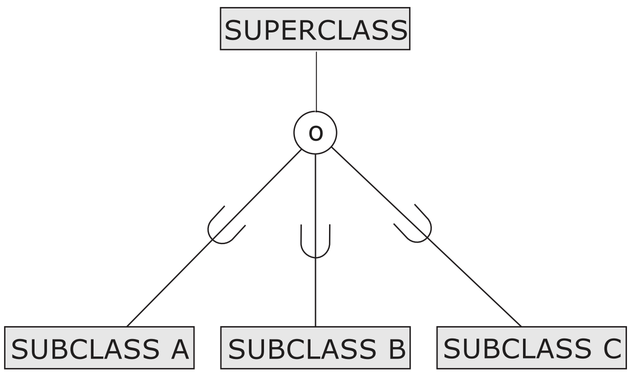


Table 8.1 Operations of Relational Algebra

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>