UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Exam

DIT033 / DAT335 – Data Management

Wednesday, June 5, 2024 08:30 - 12:30

Examiner:
Philipp Leitner

Contact Persons During Exam:
Philipp Leitner (+46 733 05 69 14, philipp.leitner@chalmers.se)

Allowed Aides:
None except English dictionary (non-electronic), pen/pencil, ruler, and eraser.

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

Grade Limits:
0 – 49 pts: U, 50 – 69 pts: 3, 70 – 84 pts: 4, 85+ pts: 5

Review:
Exams can be reviewed at student office after grading is complete.

Task 1 – Theory and Understanding (22 pts)

Each of the following questions requires an approximately two to four paragraphs long answer **in your own words**. If a question ask for an example, you should develop your own examples (simply re-using an existing example from the lecture slides or the Internet is not sufficient). Use figures or sketches if appropriate. Read the questions carefully, and make sure to answer the question that is asked.

Q1.1: What are the three transaction boundaries? Explain what they are used for. *(3 pts)*

Q1.2: What is normalization in databases? Briefly explain the three normal forms (NF1, NF2, and NF3). *(4 pts)*

Q1.3: What is an index in databases? Explain the advantages and disadvantages of indices. *(6 pts)*

Q1.4: Explain the differences between inner join and full outer join. Demonstrate the answer using an example of two tables with at least three rows. *(6 pts)*

Q1.5: What are prepared statements in SQL and why are they useful (give two benefits)? *(3 pts)*

Task 2 – EER Diagrams (24 pts)

Consider the following excerpt of the domain description for a conference 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.

Online Store:
The database needs to keep track of products. Products are identified through a product ID and name, both of which are independently unique. Products also need to store a product description, which is not necessarily unique. Finally, each product is classified into exactly one product category, and each product category contains arbitrary many products. Product categories can be formed hierarchically – that is, a product category may also contain other sub-categories (for example, the category ”milk product” may contain the category ”yoghurt”, which may contain the product ”Arla Greek Yoghurt”). A product category has an unique name, and we need to have access to how many products and other categories are in each product category.

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. Additionally, every order needs to have a status. Each order is handled by exactly one employee, but not every employee handles orders.

Finally, some employees manage product categories. Each employee can be responsible for zero to many product categories, but each category is managed by exactly one employee.

Task 3 – Mapping an EER model (14 pts)

Consider the EER model below. Construct a relational model that represents this domain as precisely as possible. 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)

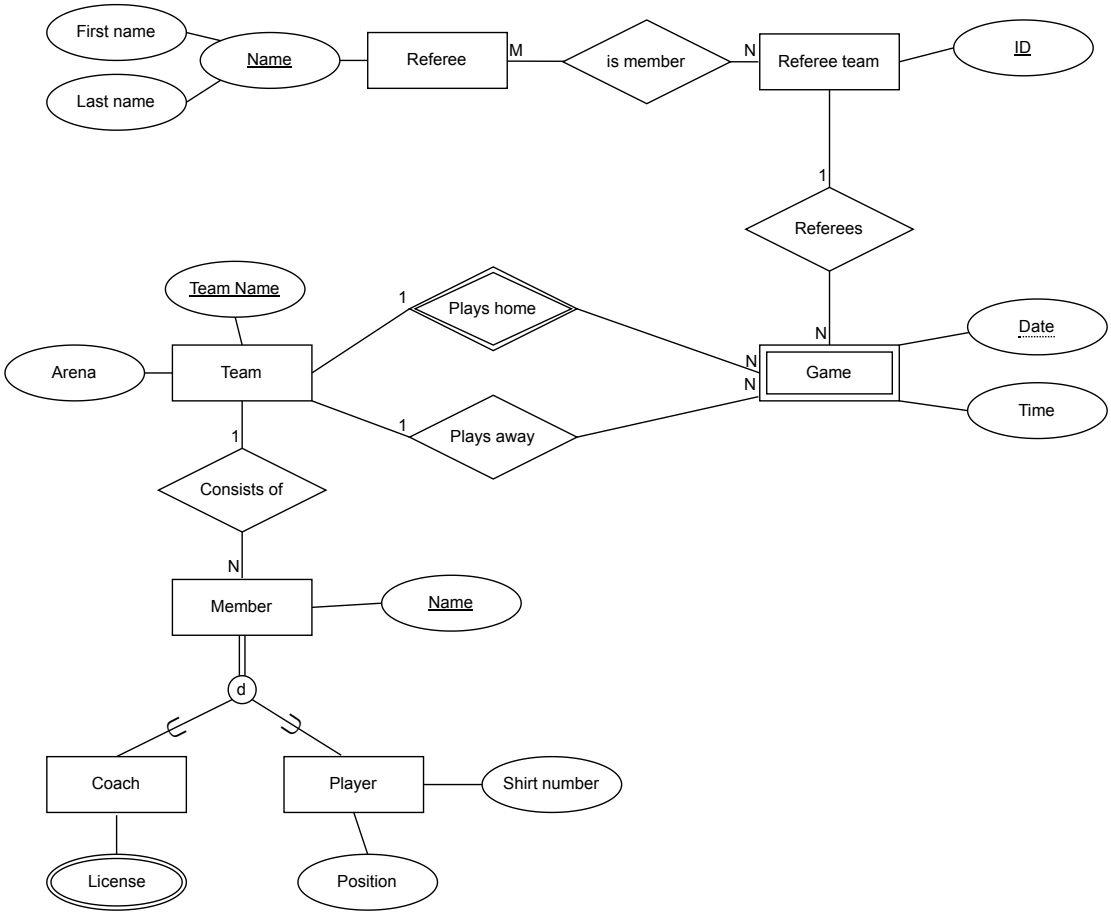


Figure 1: System to manage games in a team sport

Task 4 – Relational Algebra (20 pts)

Relational Model:

CAGE(<u>number</u>, size, type)

SPECIES(<u>id</u>, name, habitat, conservation_status)

ANIMAL(<u>id</u>, name, age, gender, cage_id, species_id)
cage_id → CAGE.id
species_id → SPECIES.id

The type of the cage can be, for example, indoor, outdoor, or aquatic. Habitat describes the natural environment of the species, and the conservation_status of the species can be, for example, endangered or vulnerable. Age is provided in years.

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 List all male animals along with their cage type and species habitat.

Q4.2 Get the details of all species that are endangered.

Q4.3 Find the number of animals in each cage.

Q4.4 Find the name, age in days, cage size, and cage type of each male animal. Some animals are not in cages, these should still be contained in the resulting tuple with a cage size and type of NULL.

Task 5 – SQL (20 pts)

Relational Model:

CAGE(number, size, type)

SPECIES(id, name, habitat, conservation_status)

ANIMAL(id, name, age, gender, cage_id, species_id)
cage_id → CAGE.id
species_id → SPECIES.id

The type of the cage can be, for example, indoor, outdoor, or aquatic. Habitat describes the natural environment of the species, and the conservation_status of the species can be, for example, endangered or vulnerable. Age is provided in years.

Assuming that this relational model has been implemented as SQL tables, write the appropriate SQL statements to perform the following tasks. All tasks shall be done with a single SQL statement.

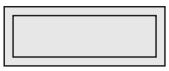
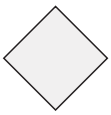
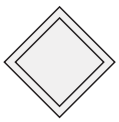
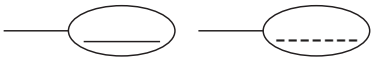
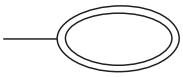
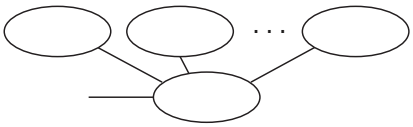
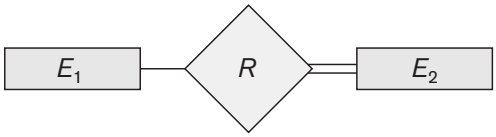
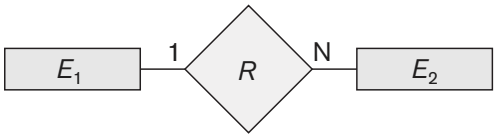
Q5.1 Create the table ANIMAL. Assume that the other two tables already exist. Values are required for the ”name” attribute, and ensure that age cannot be higher than 250. If a cage or species is deleted, all animals in this cage or of this species should also be deleted automatically.

Q5.2 Find the names and ages of all animals kept in either indoor or outdoor cages. Write a SQL query with a single condition in the WHERE clause.

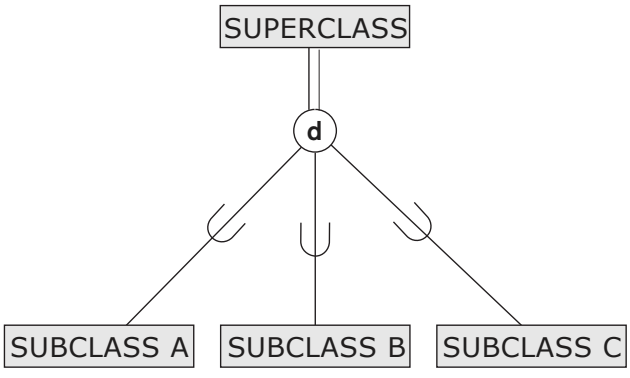
Q5.3 List all cages that do not currently house any female animals.

Q5.4 Find the id, name, and age of all animals older than the average age of this species.

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

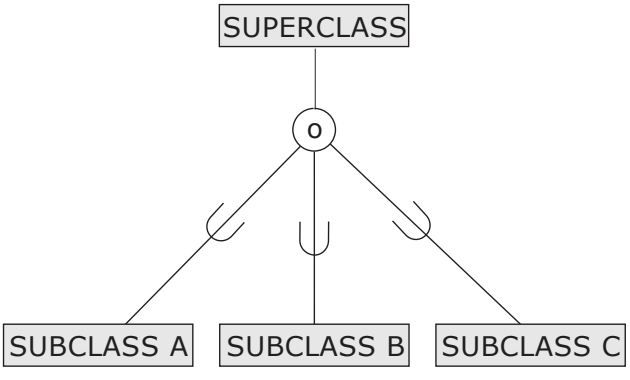


Table 8.1 Operations of Relational Algebra

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 $R_1 \bowtie_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$
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 $R_1 *_{\text{(<join attributes 1>), (<join attributes 2>)}} R_2$ 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>