



UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Exam

DIT032 / DIT033 / DAT335 – Data Management

Wednesday, August 17, 2022, 14:00 - 18:00

Examiner:

Philipp Leitner

Contact Persons During Exam:

Nayla Nasir (+46 31 772 1986)

(visitations around 15:00 and 16:30)

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 – 49.9 pts: U, 50 – 69.9 pts: 3, 70–84.9 pts: 4, 85+ pts: 5

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

Task 1 – Theory and Understanding (20 pts)

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

Q1.1: Describe the basic database operations introduced in this course (the CRUD operations). Specify an example database table and provide concrete examples in SQL for all four operations. (5 pts)

Q1.2: What are the three different strategies for mapping inheritance to the relational model? How these strategies can be used for different types of specialization (i.e Disjoint-total, Disjoint-partial, Overlapping-total and Overlapping-Partial)? (5 pts)

Q1.3: Assume a database isolation level of READ UNCOMMITTED. Provide and discuss an example database interaction that suffers from a dirty read. Develop your own example – existing examples from the lecture slides, book, or the Internet do not count. (5 pts)

Q1.4: Describe and compare replication and sharding. In what ways are they similar, and how do they differ? Provide an example of a scenario in which both sharding and replication would be useful. (5 pts)

Task 2 – EER Diagrams (20 pts)

Consider the following excerpt of the domain description for the database of a streaming service. 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 and only if something is not specified, make a reasonable assumption and note it down in plain text.

Streaming Service Database:

At the heart of the system are users. For users we need to save an unique username and their watching preferences. There are two types of users. Free users have a quota (their account is disabled if they watch more than their quota in a month). Paying users have no quota, but we need to store their credit card information, which consists of an unique credit card number, an address, and a validity date. Free user accounts can be linked to paying accounts, which increases their quota. Every free account can be linked to at most one paying account, and every paying account can have a maximum of 3 free accounts linked to it.

Users stream content, i.e., movies or series episodes. Whenever a user streams content, we need to save the actual duration of how long they watched. For simplicity, you can assume that users are unable to stream the same content multiple times, but of course the same content can be streamed to multiple users, and a user can stream multiple different content items. For all types of content, we need to save the genre, the length, and the release date. Further, we need to save the actual movie data. For movies, we also need to store an unique title. Episodes are collected in series. For each series, we need to store its unique title, when it aired for the first time. A series can have multiple episodes, each of which has a title (which is optional and does not have to be unique). Episodes are identified through an episode key consisting of the combination of season and episode number (e.g., Season 1 - Episode 5). Episode keys are only unique in combination with the title of the series.

For each user, the system generates exactly one unique personalized recommendation list. Each list is identified through an unique id, and contains 10 items, typically (but not necessarily) a combination of movies and series. Movies and series can be contained in multiple recommendation lists, but don't have to be in any.

Finally, we need to keep track of actors in our database. We need to store the (unique) names of actors, but we need to keep track of which episodes or movies they appeared in and which role they played. An actor has to have appeared in at least one episode or movie, but they could have been in many. Most streaming content has many actors appearing in them, but some also make due without any actors (e.g., animation movies). As a special feature, paying users can follow any number of actors and get informed whenever new content featuring them is released. Evidently, the same actor can be followed by many paying users.

Task 3 – Mapping EER to the Relational Model (14 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:

COURSE (shortname, fullname, syllabus)

TEACHER (id, name, salary, course)

course → COURSE.shortname

COURSE_INSTANCE (course, year, start, end)

course → COURSE.shortname

STUDENT (id, name, semester)

STUDENT_COURSEINSTANCE (course, student, year, grade)

{course, year} → {COURSE_INSTANCE.course, COURSE_INSTANCE.year}

student → STUDENT.id

(course is a foreign key pointing at COURSE.shortname, student is a foreign key pointing at STUDENT.id, and the combination of course and year in STUDENT_COURSEINSTANCE is a foreign key pointing at COURSE_INSTANCE)

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 the course short names and durations (defined as the difference between end and start date) of all course instances running in 2019.

Q4.2: Return the names of all students registered to the course with the fullname “Data Management”.

Q4.3: Find the highest salary of a teacher not teaching any courses.

Q4.4: Return a list of courses and the years they have been offered in. Each tuple in your result should contain shortname, fullname, syllabus, and the year. If a course has never been offered, it should still be contained in your result (with a NULL value for the year).

Task 5 – SQL (20 pts)

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

Queries:

Q5.1: Create the table `STUDENT_COURSEINSTANCE`. Assume that the `STUDENT` and `COURSE_INSTANCE` tables already exist. Select suitable data types. Grades should be integer values smaller than 6. If a student is deleted, all their records in `STUDENT_COURSEINSTANCE` should automatically also be deleted.

Q5.2: Return the short and full names of all courses where the syllabus contains the word “database”.

Q5.3: Return the names and salaries of all teachers of the student with the name “Anna Berg”. Avoid returning duplicate teachers, and order results from high to low salaries.


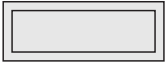
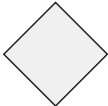
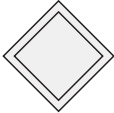

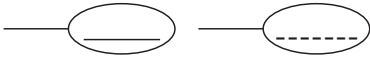

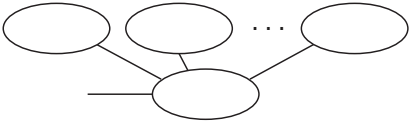

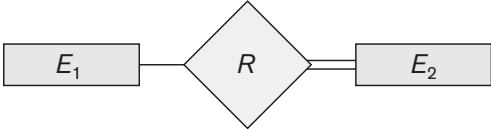
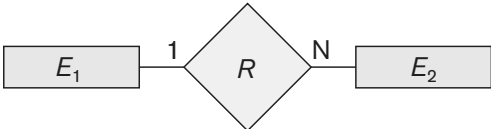
Q5.4: Return the names of all teachers whose salary is above the average teacher salary.

Task 6 – Data representation (6 pts)

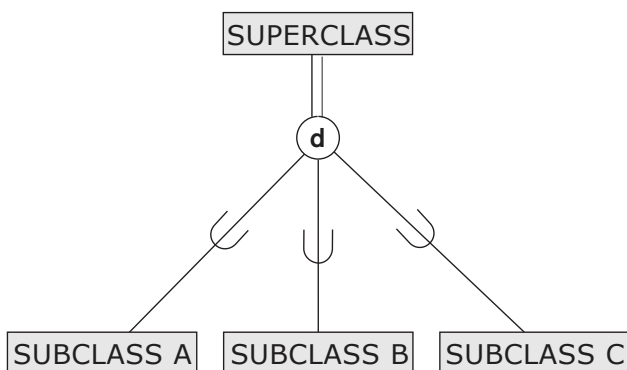
Consider the JSON document below. Provide a well-formed and logically structured XML file that captures the same information. Make use of all of the following: attributes, subelements, and text nodes.

```
{
  "id" : 146,
  "name" : "Nayla Nasir",
  "departments" : ["CSE"],
  "projects" : [
    {"id" : 14, "name" : "CONF-2"},
    {"id" : 21, "name" : "Immersed"}
  ],
  "courses" : ["DIT033", "DAT335", "DAT295"],
  "started_at" : "01.09.2020"
}
```


Appendix: Notation Guidelines for EER and RA

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying 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

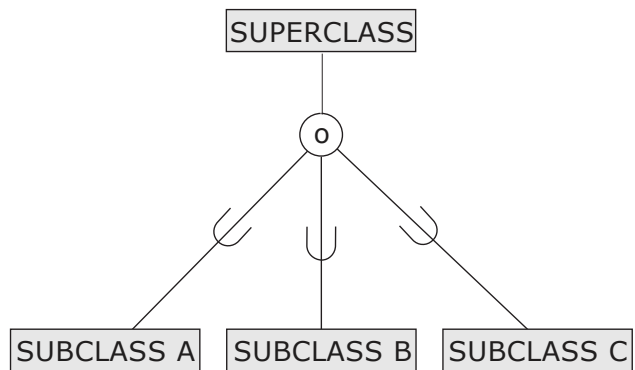


Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R .	$\sigma_{\langle \text{selection condition} \rangle}(R)$
PROJECT	Produces a new relation with only some of the attributes of R , and removes duplicate tuples.	$\pi_{\langle \text{attribute list} \rangle}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$, OR $R_1 \bowtie_{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$
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.	$R_1 \star_{\langle \text{join condition} \rangle} R_2$, OR $R_1 \star_{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$
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 $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

$\langle \text{grouping} \rangle \mathcal{F} \langle \text{functions} \rangle (R)$

whereas $\langle \text{functions} \rangle$ is a list of

$[\text{MIN} | \text{MAX} | \text{AVERAGE} | \text{SUM} | \text{COUNT}] \langle \text{attribute} \rangle$