UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Exam

DIT033 / DAT335 – Data Management

Wednesday, August 21st, 2024 14:00 - 18:00

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 (24 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: Describe the three-schema ANSI/SPARC database architecture. Name and briefly describe each of the three different schemata. *(6 pts)*

Q1.2: What is query rewriting? When do we use it in the context of data management? Provide a small example of query rewriting. *(6 pts)*

Q1.3: What are transactions in a database system? What are they primarily used for? *(4 pts)*

Q1.4: Explain two common advantages as well as two common disadvantages that NoSQL database systems often have over a relational system such as Postgres. *(4 pts)*

Q1.5: Explain the query-by-example approach that MongoDB uses for querying. *(4 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.

Twitter (also known as ”X”):
Our database needs to store Twitter users, tweets, and direct message conversations. Users have an unique username, a full name (which is composed of first and last name), and a registration date. Users subscribe to updates of other users by “following ” them. Any user can follow any number of others, and any user can have an arbitrary number of followers. We also need to have access to how many other users a user is following.

Users may send an arbitrary number of tweets (short messages). A tweet consists of a globally unique identifier, a text, and the date when the tweet has been sent. A tweet may also tag zero to many users. Finally, a tweet may be a retweet of a previous message. In that case the tweet needs to refer back to the original message. However, a single tweet may only retweet (at most) one previous tweet. For each retweet, we need to store the time stamp of the retweet.

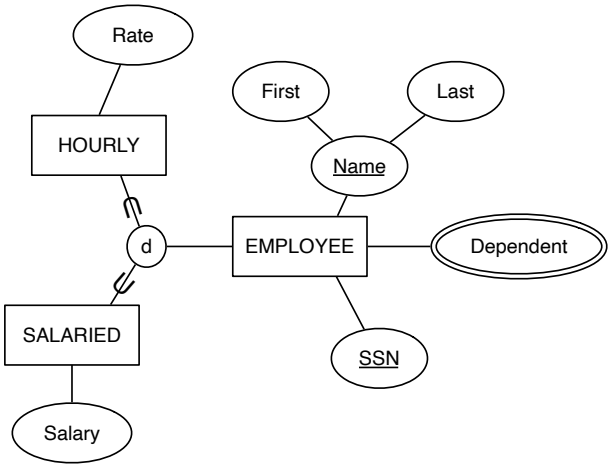
There are two special types of tweets. Firstly, replies are tweets sent in response to a previous tweet. In addition to all properties of normal tweets, replies have a thread number and point to exactly one previous tweet. Secondly, media tweets are defined as tweets that have one or more media items (e.g., pictures or videos) embedded. Tweets can at the same time be replies and/or media tweets (or neither, of course). For media items we need to store a globally unique identifier and the URL of the media item.

In addition to publicly viewable tweets, users may also have private conversations. A conversation is always between exactly two users (*Note: consider how to map this requirement using the 1/N/M notation!*), is identified through a globally unique ID, and points to both users involved in the conversation. Further, a conversation contains one to many messages. Messages are identified through a running number that is only unique in the context of a specific conversation. Further, messages contain the message text, and they may optionally also embed one or multiple media items.

Task 3 – Mapping an EER model (12 pts)

Consider the EER model below. Construct **two alternative** mappings to the relational model using the different ways to map inheritance we covered in the course. 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).

For both alternatives you provide, discuss advantages and disadvantages in your own words.



Task 4 – Relational Algebra (20 pts)

USER(<u>username</u>, quota, reg_date)

MOVIE(<u>title</u>, genre, duration)

STREAM(<u>user</u>, <u>movie</u>, date, completion)
user → USER.username
movie → MOVIE.title

ACTOR(<u>id</u>, firstname, lastname)

APPEARED_IN(<u>actor</u>, <u>movie</u>)
actor → ACTOR.id
movie → MOVIE.title

(MOVIE.duration is in minutes. STREAM.completion is a percentage – e.g., 0.75 represents 75% – indicating how much of the movie's total duration the user streamed before stopping the stream)

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 Get all details of all streams of the movie "Titanic" that have been watched for more than 20% of the movie.

Q4.2 For all actors, list their first name, last name, movie title, and movie genre for each movie them have appeared in. Actors that have not appeared in any movie should still be listed, with NULL values for movie details.

Q4.3 Generate a list of pairs of movie titles of movies in the same genre and which are approximately the same length (i.e., their duration is not more than 10% different).

Q4.4 Provide a list per date of how many minutes of movies have been streamed on this day (in total across all users and movies). You do not have to consider the corner case where a stream ends on a different day than when it started.

Task 5 – SQL (20 pts)

Assume that the relational model from Task 4 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.

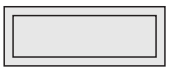
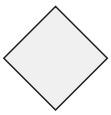
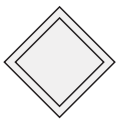
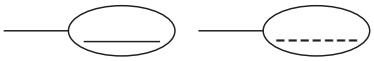
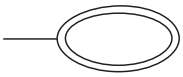
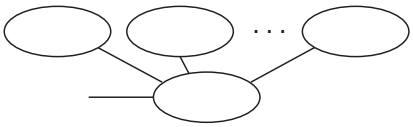
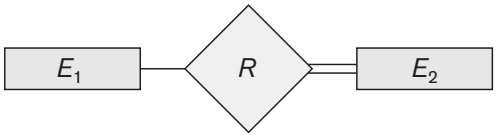
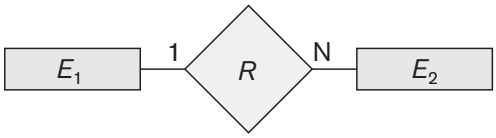
Q5.1 Create a materialized view of all actors and the movies they appeared in. Include all attributes of actors and movies in the view. Also explain what update strategy you suggest for this view, and why.

Q5.2 Find the first names of all actors with last name ”Muller”.

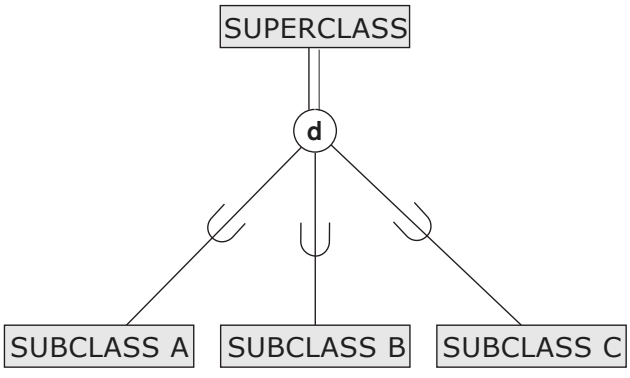
Q5.3 Find the username and registration date of all users who have never streamed a movie.

Q5.4 Find how many movies have been streamed on Christmas day 2023 (2023-12-25). Only include streams that have run for at least 10% of the movie.

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

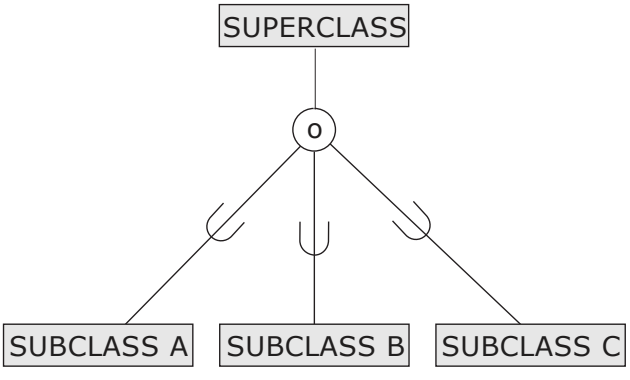


Table 8.1 Operations of Relational Algebra

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>