



UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Exam

DIT033 / DAT335 – Data Management

Monday, March 11, 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:

Exam can be reviewed at student office after the exam. Time and date of the exam review will be announced through Canvas.

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: Define the ACID properties in the context of database transactions. (*4 pts*)

Q1.2: Explain referential integrity in Entity Relationship models using an example. (*4 pts*)

Q1.3: Describe three criteria for a “good” design of a database. (*4 pts*)

Q1.4: Indicate whether each of the following statements is true or false: (*3 pts*)

1. Functional dependencies are bi-directional.
2. Projection is typically cheap to execute.
3. Weak relationships can only exist with weak entities.
4. It is not possible to update a view if it is based on a join.
5. In JDBC, we need to open a new connection to execute each statement.
6. NoSQL databases are usually used for structured data.

Q1.5: What can we do to improve slow queries? Explain two alternatives. (*4 pts*)

Q1.6: Suppose a database has the isolation level “READ COMMITTED”. What does that entail and what possible violations can occur and why? How can we eliminate these violations? (*5 pts*)

Task 2 – EER Diagrams (22 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.

Conference Management:

The database needs to keep track of conferences. Conferences have an unique name, a location, a duration, and an edition (e.g., 2nd edition). The duration is specified through a begin and end date. We also want to track how many conferences there are in total in the system.

Conferences are attended by delegates (i.e., people attending the conference). Delegates can attend multiple conferences, and clearly a conference can have many delegates attending. For each delegate, we need to store when they registered for each conference (“reg_date”). For each delegate, we further need to store their registration number (which is unique for each delegate), their name, country, and date of birth.

Conferences further consist of tracks. Tracks have a name which is unique for a conference (but not necessarily unique overall), a date, start time, and end time. Every track has exactly one track chair, who is a special delegate responsible for coordinating the track. For delegates that are also track chairs, we need to save the organisation that they represent in the conference (this is not needed for regular delegates). A track chair can chair multiple tracks, but they need to chair at least one to be considered a track chair.

Finally, tracks consist of presentations. For each presentation, we need to save the title, which is unique in a track (but, again, not necessarily unique overall), and the start time. Each presentation is given by exactly one delegate (the presenter), but there may be delegates who are not giving any presentations. Each delegate may give at most one presentation.

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)

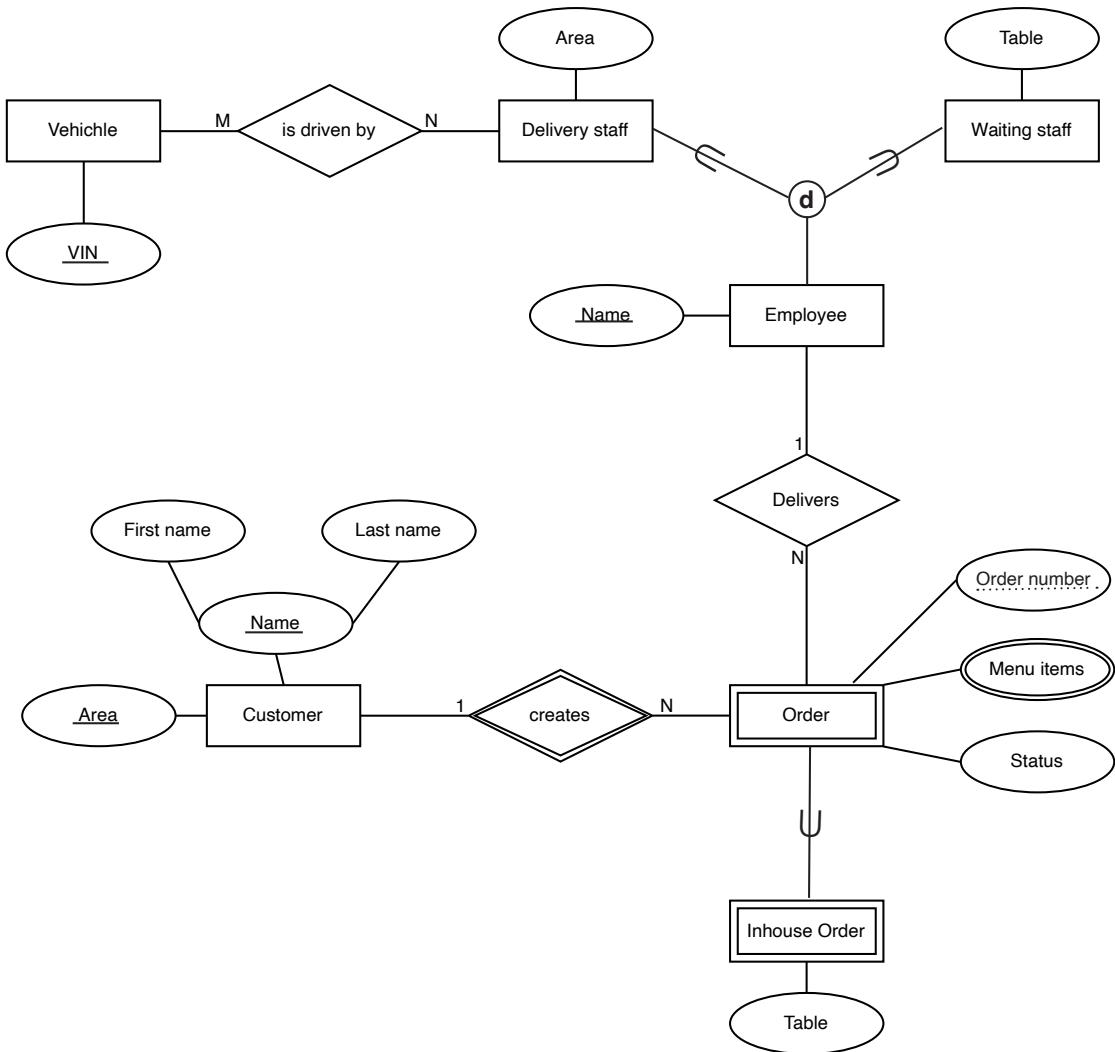


Figure 1: Restaurant order and delivery system

Task 4 – Relational Algebra (20 pts)

Relational Model:

ARTIST(name, bio)

ALBUM(title, release_date, genre, creator)
creator -> ARTIST.name

SONG(title, duration, artist, album)
{artist, album} -> {ARTIST.name, ALBUM.title}

The duration of each song is represented in seconds. ALBUM has one foreign key - creator points at ARTIST.name. SONG has two foreign keys - one pointing at ARTIST.name and one pointing at ALBUM.title.

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 titles, artist names, and durations of all songs longer than 3 minutes (180 seconds).

Q4.2 Find the name and biography of the artist who created the song titled 'Shape of My Heart'.

Q4.3 List the titles of the other songs that are on the same album as 'Shape of My Heart'.

Q4.4 Calculate the total duration of the album containing the song 'Shape of My Heart' in the format of minutes.

Task 5 – SQL (20 pts)

Relational Model:

ARTIST(name, bio)

ALBUM(title, release_date, genre, creator)
creator -> ARTIST.name

SONG(title, duration, artist, album)
{artist, album} -> {ARTIST.name, ALBUM.title}

The duration of each song is represented in seconds. ALBUM has one foreign key - creator points at ARTIST.name. SONG has two foreign keys - one pointing at ARTIST.name and one pointing at ALBUM.title.

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 query.

Q5.1 Retrieve song title, artist name, album title, duration, release date, and artist bio of each song. A song's release date is the same as the release date of the album they are in.

Q5.2 Retrieve all songs by 'Taylor Swift' along with all album details that have been released after 2020, sorted by release date.

Q5.3 Create a (non-materialized) view that lists the names of all artists and how many albums they have released (call this column 'NrAlbums'). Only include artists that have at least 3 albums.

Q5.4 Delete the artist with the name 'Taylor Swift', and all their albums and songs.
You can assume that all foreign keys in this model are configured with 'ON DELETE CASCADE'.

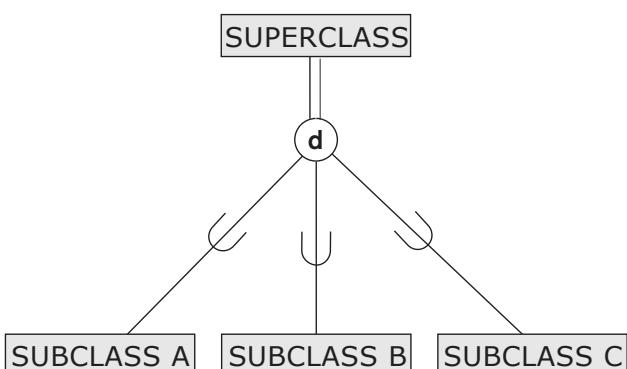
Q5.4 Delete the artist with the name 'Taylor Swift', and all their albums and songs.

You can assume that all foreign keys in this model are configured with 'ON
DELETE CASCADE'.

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

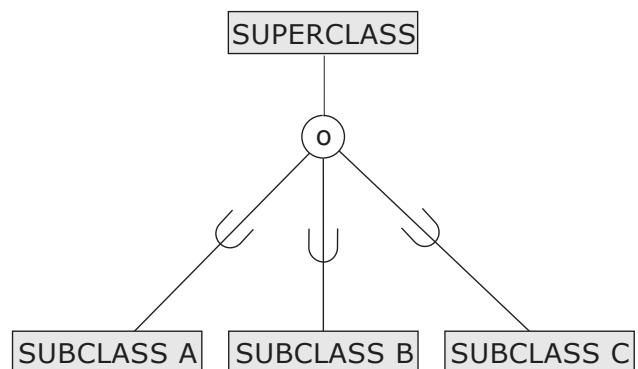


Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R .	$\sigma_{\text{selection condition}}(R)$
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 $R_1 *_{(\text{join attributes 1}), (\text{join attributes 2})} R_2$ $R_2 \text{ OR } R_1 * R_2$
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 $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

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

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

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