UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Exam

DIT034 / DAT335 – Data Management

Monday, March 17, 2025 08:30 - 12:30

Examiner:
Philipp Leitner

Contact Persons During Exam:
Mazen Mohamad (:+46 73 023 93 76, mazenm@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.

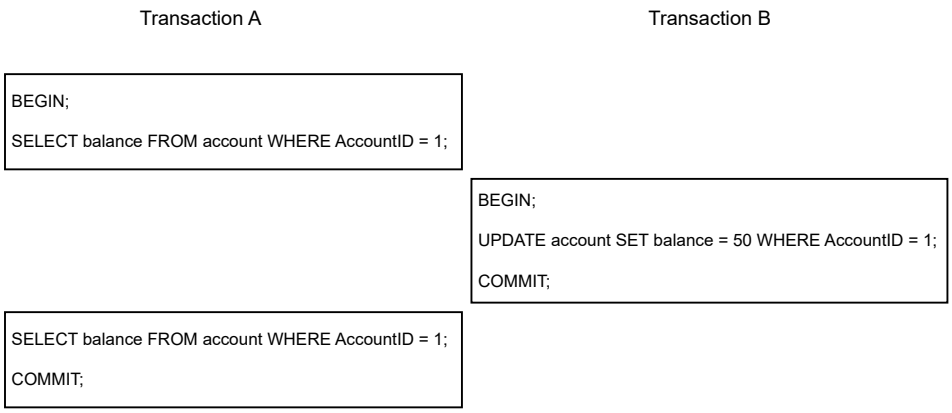
Task 1 – Theory and Understanding (22 pts)

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 constraints can be defined for relationships in Entity-Relationship (ER) Models? Provide a simple example for each constraint to illustrate its use. *(4 pts)*

1. **Cardinality** (1:1, 1:N, N:1, M:N) (1pt)
A student can enroll in multiple courses, but each course has many students (many-to-many). (1pt)
2. **Participation constraint** (1pt) (Required, Not Required)
Every employee must be assigned to at least one department (total participation). (1pt)

Q1.2: In the example below, we have two parallel transactions (Transaction A and B). The isolation level is set to READ COMMITTED. Discuss what isolation violation occurs in this example and how it can be mitigated. *(4 pts)*



1. The isolation violation is **Non-repeatable reads** (1pt) which occurs when reading the same data multiple times within a transaction, but another transaction modifies or deletes that data before the initial transaction is completed, leading to **inconsistent results within the same transaction of the same reads**. (2pt)

2. It can be mitigated by changing the isolation level to **Repeatable reads** or **Serializable** (1pt).

Q1.3: (i) Identify one issue with this database design below and indicate (ii) which normal form this database design violates, and the reason for the violation? *(4 pts)*

TEAM (TeamID, Name, Sport) PLAYER (PlayerID, Name, BirthDate, Team, TeamName) Team → TEAM.TeamID

1. Issue in the database design (2pt): Unwanted functional dependency (redanduncy)
2. Normal form violated (1pt): Third normal form
3. Reason for violation (1pt): non-key attribute has a functional dependency on another non-key attribute. More specifically, TeamName has a functional dependency with Team (**Team implies TeamName**) which is **redundant**.

Q1.4: What is query rewriting (query optimization) in databases? Explain each type of query rewriting briefly? *(5 pts)*

1. Query rewriting (query optimization) is the process of finding the best query plan that is semantically equivalent to the original query. (1pt)
2.Heuristic optimization (1pt): Apply general RA equivalence rules to convert queries into new query plans that we know are, in general and independently of the concrete database, faster (1pt)
3. Cost-based optimization (1pt): Estimate the "cost" (duration) of each operation based on the actual data in the database. Execute the query plan that has the lowest cost. (1pt)

Q1.5: Explain the main functions in MapReduce in text (you may also visualize the process) and discuss two of its advantages. *(5 pts)*

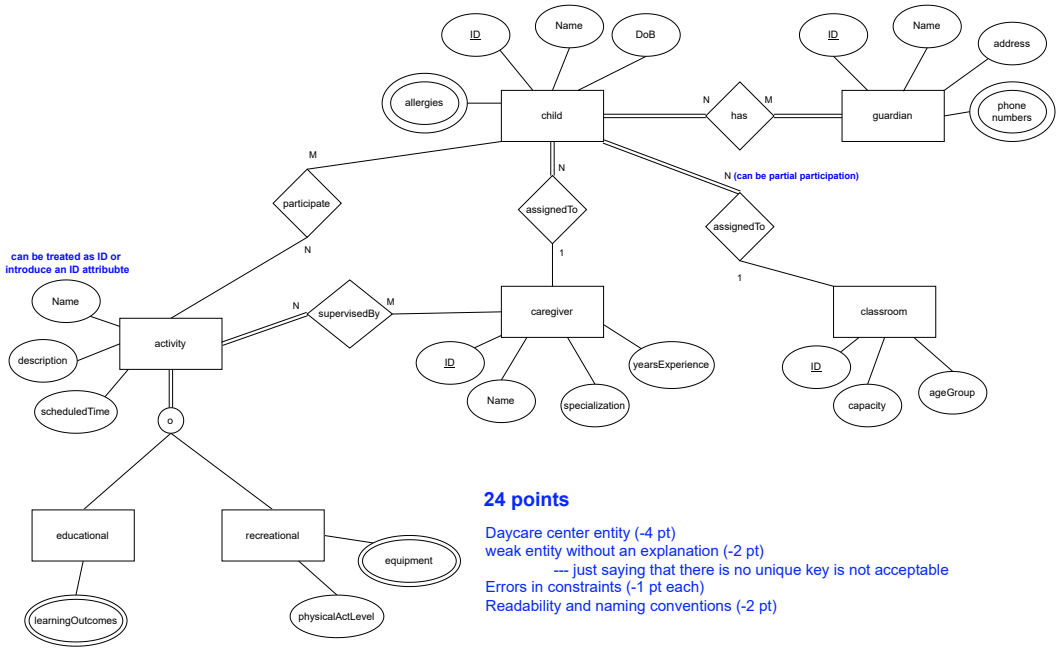
- Map function: it takes items from a datastore, and emits/returns key-value pairs of each item (1pt)
- Reduce function: it takes a key and (some or all) the values of that key, then returns/emits a new key-value pair (1pt)
- Advantages (any of two of these): (2pt)

Fault-tolerance, parallelizable, sharding-ready (+1 pt for details)

Task 2 – EER Diagrams (24 pts)

Consider the following excerpt of the domain description for a daycare 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. If you think something is not specified, make a reasonable assumption and write it down in plain text.

Daycare center
A daycare center needs a database to manage its daily operations efficiently. The system should store information about children, parents/guardians, caregivers, classrooms, and activities. Your task is to design an Entity-Relationship Diagram (ERD) for the daycare management system based on the following requirements:
Children are enrolled in the daycare. Each child has a unique ID, name, date of birth, and may have multiple allergies (if any). Each child must have at least one registered parent/guardian but can have multiple. A parent/guardian can be responsible for multiple children. Parents/guardians have a unique ID, name, phone number(s) (a parent can have multiple contact numbers), and address. The daycare employs caregivers who take care of the children. Each caregiver has a unique ID, name, specialization (e.g., infant care, toddler care), and years of experience. Each child is assigned to exactly one caregiver, but a caregiver can be responsible for multiple children. The daycare has multiple classrooms. Each classroom has a unique ID, a capacity, and is designated for a specific age group. A child is assigned to one classroom at a time, but a classroom can have multiple children (up to its capacity).
The daycare organizes activities. Each activity has a name, description, and a scheduled time. There are two types of activities, educational activities and recreational activities, an activity can be both or either types. Educational activities have learning outcomes, while recreational activities have list of equipment needed and the physical activity level.
Children can participate in multiple activities, and each activity involves multiple children. Each activity is supervised by at least one caregiver, and a caregiver may supervise multiple activities.



24 points

Daycare center entity (-4 pt)
weak entity without an explanation (-2 pt)
--- just saying that there is no unique key is not acceptable
Errors in constraints (-1 pt each)
Readability and naming conventions (-2 pt)

Task 3 – Mapping an EER model (14 pts)

Consider the EER model below representing a system that keeps track of board game sessions. 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).

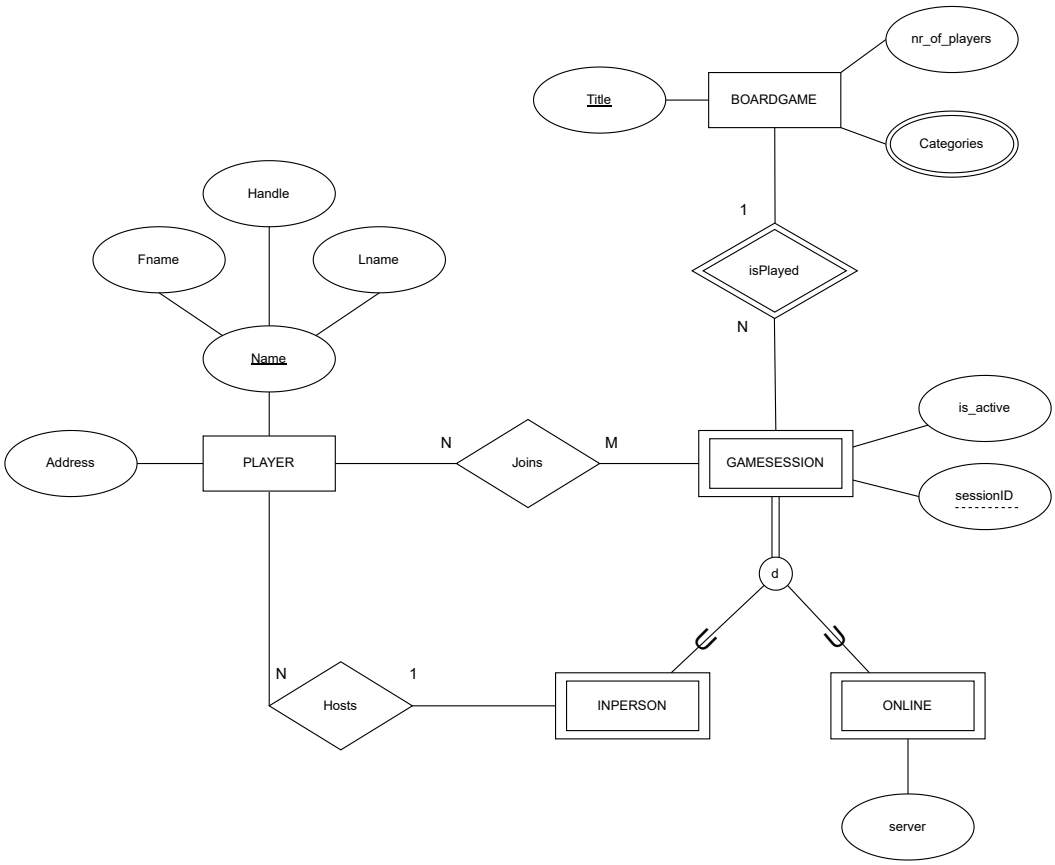


Figure 1: System to coordinate board game sessions

For all you can always introduce a separate ID as Key, if things gets to messy. The reason for a game session to be present in the PLAYER is that we swapped the N-1 in hosts. Should have been a 1-N, and then a person would be in in-person as the host. (made more sense human logically, but it's still mappable)

PLAYER(Name, Address, Session_id, Title)

Name → NAME.Id

Title → GAMESESSION.Title

Session_id → GAMESESSION.Session_id

Name of player is a Composite attribute, so you can add fname, handle, lname to PLAYER but then the composite attribute is a composite Key (all three has to be keys) You cannot have name, fname handle lname in the same.

NAME (ID, Fname, Handle, Lname)

BOARDGAME (Title, nr_of_players)

Multi valued attributes require a table of its own.

Categories (Title, Category)

Title → BOARDGAME.title

Also ok ofc to insert artificial key

Categories (ID, Title, Category)

Title → BOARDGAME.title

disjoint total. Gamesession has to be either in person or online, which means you can scrap the "gameSession" table entirely and only have the sub-types. But then is_active has to be present in the sub-tables and you need two "JOINS", one for each subtype. Gamesession is weak entity, can't be defined without the board game name.

GAMESESSION(session_id, Title, is_Active)

Title → BOARDGAME.title

INPERSON_GAMESESSION(session_id, Title)

Title → GAMESESSION.title

Session_id → GAMESESSION.session_id

ONLINE_GAMESESSION(session_id, Title, Server)

Title → GAMESESSION.title

Session_id → GAMESESSION.session_id

Since multiple players can participate in multiple sessions this needs a separate table:

Player_joins_Gamesession(name, title, session_id)

Name → player.name

Title → GAMESESSION.title

Session_id → GAMESESSION.session_id

14p total 1p Boardgame, Name handled correctly, player correctly, hosts correctly in Players.

2p Multi valued attribute Categories (being own table, plus 2 keys)

3p Many to many Joins (own + correct keys)

5p Weak entity GameSession (1+correct keys tot 3p), plus disjoint Total participation (so is_active placement plus two or one JOINS) 2p

Task 4 – Relational Algebra (20 pts)

Relational Model:
FLIGHT (id, departure_airport, arrival_airport, departure_time, arrival_time, aircraft)
aircraft → AIRCRAFT.id
AIRCRAFT(id, model, capacity)
PASSENGER(id, first_name, last_name, email)
BOOKING(id, flight, passenger, seat_number)
flight → FLIGHT.id
passenger → PASSENGER.id

The departure and arrival times are represented in a datatype DATE. Flight has one foreign key that points to aircraft ID. Booking has two foreign keys, flight and passenger, that point to the flight ID and passenger ID.
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.

Q4.1: List all aircrafts that have a capacity of more than or equal to 200 passengers. *(5 pts)*

$\sigma_{capacity \geq 200}(AIRCRAFT)$

or equivalently:

$\sigma_{capacity > 199}(AIRCRAFT)$

Q4.2: Get the model of the aircrafts that will land at Landvetter airport between 1/4/2025 and 1/5/2025. *(5 pts)*

$\pi_{model}(\sigma_{arrival\_airport='Landvetter' \wedge arrival_time \geq '2025-04-01' \wedge} \sigma_{arrival\_time < '2025-05-01'}$
$(FLIGHT \bowtie_{FLIGHT.aircraft=AIRCRAFT.id} AIRCRAFT))$

Q4.3: Return all the destinations that *Philipp Leitner* has booked in February 2025. *(5 pts)*

$\pi_{arrival\_airport}(\sigma_{first\_name='Philipp' \wedge last\_name='Leitner' \wedge}$
$departure\_time \geq '2025-02-01' \wedge departure\_time < '2025-03-01'$
$(PASSENGER \bowtie_{PASSENGER.id=BOOKING.passenger} BOOKING$

$\bowtie_{BOOKING.flight=FLIGHT.id} FLIGHT))$

Q4.4: Count the number of available seats per flights going to Arlanda airport with an airplane of model A320. *(5 pts)*

For each flight going to Arlanda airport using an A320 aircraft, calculate the number of available seats by taking the aircraft capacity and subtracting the number of bookings already made for that flight.

$\pi_{flight\_id,(capacity-count\_bookings)} \rightarrow available\_seats($
$\gamma_{FLIGHT.id \rightarrow flight\_id,capacity,count(BOOKING.id) \rightarrow count\_bookings}(\sigma_{arrival\_airport='Arlanda'}($
$(FLIGHT \bowtie_{FLIGHT.aircraft=AIRCRAFT.id} (\sigma_{model='A320'}(AIRCRAFT)))$
$\rtimes_{FLIGHT.id=BOOKING.flight} BOOKING)))$

It is better to use the left outer join with BOOKING (to count even flights with zero bookings). If we used an inner join, flights with no bookings (which have the maximum available seats) would be excluded from the results.

Task 5 – SQL (20 pts)

Relational Model:
FLIGHT (id, departure_airport, arrival_airport, departure_time, arrival_time, aircraft)
aircraft → AIRCRAFT.id
AIRCRAFT(id, model, capacity)
PASSENGER(id, first_name, last_name, email)
BOOKING(id, flight, passenger, seat_number)
flight → FLIGHT.id
passenger → PASSENGER.id

Assume that the relational model above is implemented in a relational database except for the relation FLIGHT.

Q5.1: Create the table FLIGHT. The departure and arrival times shall be set as Integers. The arrival_time must be after the departure_time. The departure_airport and arrival_airport cannot be the same. Values must exist for all attributes. *(5 pts)*

```
CREATE TABLE FLIGHT (
id INTEGER PRIMARY KEY,
departure_airport VARCHAR(50) NOT NULL,
arrival_airport VARCHAR(50) NOT NULL,
departure_time TIMESTAMP NOT NULL,
arrival_time TIMESTAMP NOT NULL,
aircraft INTEGER NOT NULL,
CONSTRAINT fk_aircraft FOREIGN KEY (aircraft) REFERENCES AIRCRAFT(id),
CONSTRAINT check_different_airports CHECK (departure_airport != arrival_airport),
CONSTRAINT check_valid_times CHECK (arrival_time > departure_time)
);
```

Now that the whole relational model is implemented in a relational database. Write SQL statements for the following:

Q5.2: List all arrival airports for the passenger *Philipp Leitner* along with how many times *Philipp Leitner* arrived in these airports. Order the results alphabetically based

on the airport's name. *(5 pts)*

Solution 1 (Join approach):

```
SELECT
    F.arrival_airport,
    COUNT(*) AS arrival_count
FROM
    PASSENGER P
JOIN
    BOOKING B ON P.id = B.passenger
JOIN
    FLIGHT F ON B.flight = F.id
WHERE
    P.first_name = 'Philipp'
    AND P.last_name = 'Leitner'
GROUP BY
    F.arrival_airport
ORDER BY
    F.arrival_airport ASC;
```

Solution 2 (Subquery approach):

```
SELECT
    F.arrival_airport,
    COUNT(*) AS arrival_count
FROM
    FLIGHT F
WHERE
    F.id IN (
        SELECT B.flight
        FROM BOOKING B
        WHERE B.passenger IN (
            SELECT P.id
            FROM PASSENGER P
```

```
            WHERE P.first_name = 'Philipp'
            AND P.last_name = 'Leitner'
        )
    )
GROUP BY
    F.arrival_airport
ORDER BY
    F.arrival_airport ASC;
```

Solution 3 (HAVING approach):

```
SELECT
    F.arrival_airport,
    COUNT(*) AS arrival_count
FROM
    PASSENGER P
JOIN
    BOOKING B ON P.id = B.passenger
JOIN
    FLIGHT F ON B.flight = F.id
GROUP BY
    F.arrival_airport, P.first_name, P.last_name
HAVING
    P.first_name = 'Philipp'
    AND P.last_name = 'Leitner'
ORDER BY
    F.arrival_airport ASC;
```

Q5.3: Find the number of bookings *Philipp Leitner* made in the year 2024. *(5 pts)*

```
SELECT
    COUNT(*) AS booking_count
FROM
    PASSENGER P
```

```
JOIN
    BOOKING B ON P.id = B.passenger
JOIN
    FLIGHT F ON B.flight = F.id
WHERE
    P.first_name = 'Philipp'
    AND P.last_name = 'Leitner'
    AND F.departure_time >= '2024-01-01'
    AND F.departure_time <= '2024-12-31';
```

Q5.4: Find the first and last names of the passengers that have had at least one booking in every aircraft model. *(5 pts)*

Solution 1 (GROUP BY and HAVING):

```
SELECT
    P.first_name,
    P.last_name
FROM
    PASSENGER P
JOIN
    BOOKING B ON P.id = B.passenger
JOIN
    FLIGHT F ON B.flight = F.id
JOIN
    AIRCRAFT A ON F.aircraft = A.id
GROUP BY
    P.id, P.first_name, P.last_name
HAVING
    COUNT(DISTINCT A.model) =
    (SELECT COUNT(DISTINCT model) FROM AIRCRAFT);
```

Solution 2 (Subquery approach):

```
SELECT
```

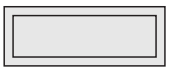
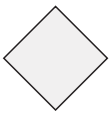
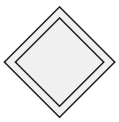
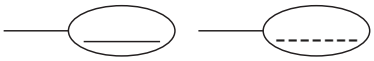
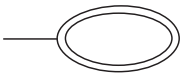
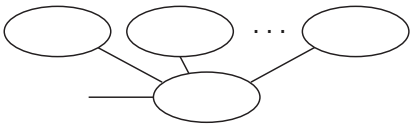
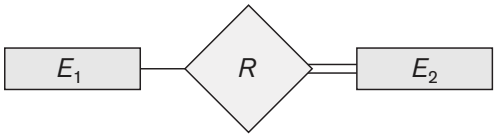
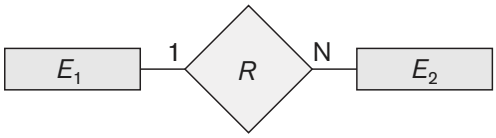
```
    P.first_name,
    P.last_name
FROM
    PASSENGER P
WHERE (
    SELECT COUNT(DISTINCT A.model)
    FROM BOOKING B
    JOIN FLIGHT F ON B.flight = F.id
    JOIN AIRCRAFT A ON F.aircraft = A.id
    WHERE B.passenger = P.id
) = (SELECT COUNT(DISTINCT model) FROM AIRCRAFT);
```

Solution 3 (Double NOT EXISTS):

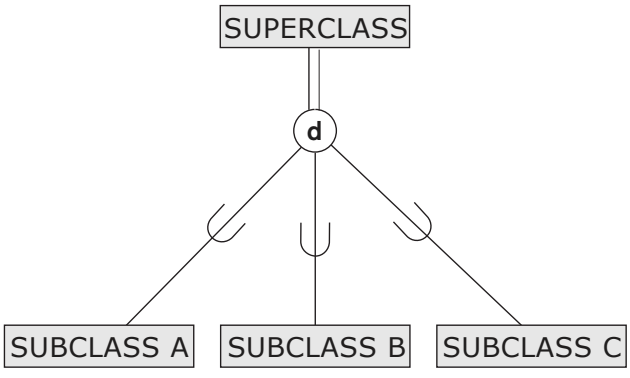
```
SELECT
    P.first_name,
    P.last_name
FROM
    PASSENGER P
WHERE NOT EXISTS (
    SELECT A.model
    FROM AIRCRAFT A
    WHERE NOT EXISTS (
        SELECT 1
        FROM BOOKING B
        JOIN FLIGHT F ON B.flight = F.id
        WHERE B.passenger = P.id
        AND F.aircraft IN (
            SELECT A2.id
            FROM AIRCRAFT A2
            WHERE A2.model = A.model
        )
    )
);
```

It uses a double negative approach: ”There does not exist an aircraft model that the passenger has not booked,” which is equivalent to ”The passenger has booked all aircraft models.”

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

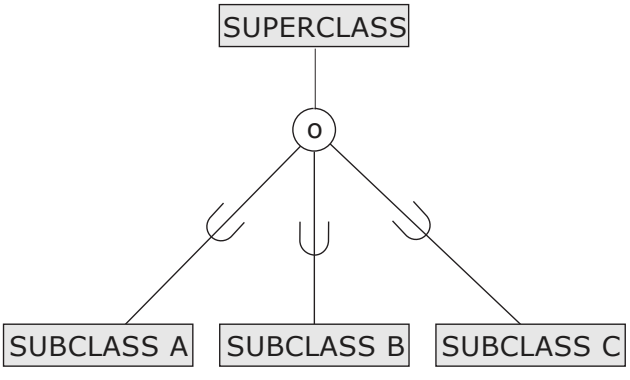


Table 8.1 Operations of Relational Algebra

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>